430 Rec'd PCT/PTO 30 AUG 1999

FORM PTO-1390 (REV 11-98) U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	ATTORNEY'S DOCKET NUMBER 6206
TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371	U.S. APPLICATION NO. (If known, see 37 CFR 1.5) 09/380250

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/FR98/02886	28 December 1998	30 December 1997

TITLE OF INVENTION METHOD FOR ASSISTING THE ADMINISTRATION OF A DISTRIBUTED APPLICATION BASED ON A BINARY CONFIGURATION FILE IN A COMPUTER SYSTEM

APPLICANT(S) FOR DO/EO/US
Christian BAILLIF and Mama Saidou DIA

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. [✓] This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. [ ] This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. [✓] This express request to begin national examination procedures (35 U.S.C. 371(f) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C. 371(b) and PCT Articles 22 and 39(1).
4. [✓] A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. [✓] A copy of the International Application as filed (35 U.S.C. 371(c)(2))
   a. [ ] is transmitted herewith (required only if not transmitted by the International Bureau).
   b. [✓] has been transmitted by the International Bureau.
   c. [ ] is not required, as the application was filed in the United States Receiving Office (RO/US).
6. [✓] A translation of the International Application into English (35 U.S.C. 371(c)(2)).
7. [ ] Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371(c)(3))
   a. [ ] are transmitted herewith (required only if not transmitted by the International Bureau).
   b. [ ] have been transmitted by the International Bureau.
   c. [ ] have not been made; however, the time limit for making such amendments has NOT expired.
   d. [ ] have not been made and will not be made.
8. [ ] A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
9. [✓] An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)).
10. [ ] A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 11. to 16. below concern document(s) or information included:

11. [✓] An Information Disclosure Statement under 37 CFR 1.97 and 1.98. cited references & Form 1449
12. [✓] An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included. to BULL S.A.
13. [✓] A FIRST preliminary amendment.
    [ ] A SECOND or SUBSEQUENT preliminary amendment.
14. [ ] A substitute specification.
15. [ ] A change of power of attorney and/or address letter.
16. [✓] Other items or information: Verification of Translation
    Cys of PCT FORMS: PCT/RO/101 and PCT/IB/301 and 308, Demande
    early receipt of S.N. post card & post card receipt
    Drawings (7) formal

09380250 . 083099

514 Rec'd PCT/PTO 30 AUG 1999

U.S. APPLICATION NO. (if known, see 37 CFR 1.5)	INTERNATIONAL APPLICATION NO.	ATTORNEY'S DOCKET NUMBER
09/380250	PCT/FR98/02886	6206

17. [✓] The following fees are submitted:

BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)) :

	CALCULATIONS	PTO USE ONLY
Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO and International Search Report not prepared by the EPO or JPO **$970.00**		
International preliminary examination fee (37 CFR 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO....... **$840.00**		
International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO **$760.00**		
International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4)........... **$670.00**		
International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4)................ **$96.00**		
ENTER APPROPRIATE BASIC FEE AMOUNT =	$ 840.00	
Surcharge of **$130.00** for furnishing the oath or declaration later than [ ] 20 [ ] 30 months from the earliest claimed priority date (37 CFR 1.492(e)).	$	

CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	18 - 20 =	0	X **$18.00**	$	
Independent claims	1 - 3 =	0	X **$78.00**	$	
MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+ **$260.00**	$	

TOTAL OF ABOVE CALCULATIONS =	$ 840.00	
Reduction of 1/2 for filing by small entity, if applicable. A Small Entity Statement must also by filed (Note 37 CFR 1.9, 1.27, 1.28).	$	
SUBTOTAL =	$ 840.00	
Processing fee of **$130.00** for furnishing the English translation later than [ ] 20 [ ] 30 months from the earliest claimed priority date (37 CFR 1.492(f)). +	$	
TOTAL NATIONAL FEE =	$ 840.00	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31). **$40.00** per property +	$ 40.00	
TOTAL FEES ENCLOSED =	$ 880.00	
	Amount to be: refunded	$
	charged	$

a. [✓] A check in the amount of $880.00 to cover the above fees is enclosed.

b. [ ] Please charge my Deposit Account No.________ in the amount of $________ to cover the above fees. A duplicate copy of this sheet is enclosed.

c. [✓] The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 11-0610 . A duplicate copy of this sheet is enclosed.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:

Edward J. Kondracki
KERKAM, STOWELL, KONDRACKI &CLARKE P.C.
Two Skyline Place, Suite 600
5203 Leesburg Pike
Falls Church, VA 22041

[illegible]
SIGNATURE:

Edward J. Kondracki
NAME

20,604
REGISTRATION NUMBER

09380250

09/380250
514 Rec'd PCT/PTO 30 AUG 1999

Docket 6206
BULL 3587HD

IN THE UNITED STATES DESIGNATED/ELECTED OFFICE (D.O./E.O./US)

Applicant:	Christian BAILLIF ET AL.
International Application No.:	PCT/FR98/02886
International Filing Date:	28 December 1998
U.S. Serial No.:	To Be Assigned
U.S. Filing Date:	August 30, 1999
For:	"METHOD FOR ASSISTING THE ADMINISTRATION OF A DISTRIBUTED APPLICATION BASED ON A BINARY CONFIGURATION FILE IN A COMPUTER SYSTEM"

Falls Church, Virginia

PRELIMINARY AMENDMENT

Honorable Commissioner of Patents
and Trademarks
Washington, D.C. 20231

Sir:

Please amend the subject application, filed concurrently herewith, as indicated below:

IN THE SPECIFICATION:

After the title and before the first paragraph on page 1, insert the following headings:

–BACKGROUND OF THE INVENTION

--FIELD OF THE INVENTION--;

Page 1, after the first paragraph and before the second paragraph at line 11, insert the following heading at the left-hand margin:

--DESCRIPTION OF RELATED ART--;

Page 2, line 4, delete "processes (BBL)" and substitute –bulletin board liaison (BBL) processes–;

Page 2, line 7, before "machine", insert –slave–;

Page 2, line 8, before "machine", insert –slave–;

Page 2, line 8, before "process", insert –bulletin board liaison– and after "process", delete "called";

Page 2, line 9, before "The bridge", insert a paragraph break.

Page 4, at line 17, before the paragraph beginning "The object..." , insert the following heading at the left-hand margin:

--SUMMARY OF THE INVENTION--;

Page 6, at line 12, and before the paragraph beginning " Other characteristics...", insert the following heading at the left-hand margin:

--BRIEF DESCRIPTION OF THE DRAWINGS--;

Page 7, before line 8, and before the paragraph beginning "The following ...", insert the following heading at the left hand margin:

--DESCRIPTION OF THE PREFERRED EMBODIMENT(S)--;

Page 7, line 11, change "six" to –seven–;

Page 7, line 12, before "network", insert –routing and–;

Page 27, delete lines 32 and 33 in their entirety, and substitute the following new paragraph:

--While this invention has been described in conjunction with specific embodiments thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the preferred embodiments of the invention as set forth herein, are intended to be illustrative, not limiting. Various changes may be made without departing from the spirit and scope of the invention as set forth herein and defined in the claims.--

IN THE CLAIMS:

Please cancel claims 1 - 12 in their entirety and without prejudice and substitute the following new claims:

--13. A process for assisting in the administration of a distributed application of a transaction processing manager, based on a binary configuration file (TUXCONFIG), characterized in that said process comprises:

- retrieving information related to said distributed application in a configuration file of a master machine (Mm), and

- checking the consistency of said application running on a given machine.

14. A process according to claim 13, characterized in that it further comprises a step for managing at least one listener module (3) of any machine of the application from another machine.

15. A process according to claim 13, characterized in that it further comprises extracting directly from the active configuration file of the master machine information related to said distributed application.

16. A process according to claim 13, characterized in that the step for checking the consistency of said application consists of comparing the information obtained from the configuration file of the master machine and the information obtained from said current application running on a given machine.

17. A process according to claim 14, characterized in that said administration of listener modules consists of starting and stopping at least one listener module,

displaying information related to at least one listener module, changing the log of at least one listener module, checking the script of at least one listener module and/or updating the script of at least one listener module.

18. A process according to claim 14, characterized in that it further comprises a step for starting and stopping a listener module running on a first machine, said step for starting and stopping being carried out by an administrator using a second machine distinct from first machine, but belonging to the same network as the first machine.

19. A process according to claim 14, characterized in that it further comprises a step for simultaneously activating several listener modules.

20. A process according to claim 14, characterized in that it further comprises a step for decompiling the active configuration file of the master machine.

21. A process according to claim 14, including a graphical interface comprising at least one icon, at least one menu and at least one dialog box for implementing the start and stop of a listener module and the retrieval of information and checking the consistency of said application running on a given machine.

22. A process according to claim 21, characterized in that the menus of the graphical interface are structured in tree form and the activation of a menu results in a display of a list of values of the current configuration, selectable by the user.

23. A process according to claim 16, further including automatically generating a file containing information on said application running on a given machine (tlog) when the file does not exist in a given machine in order to be able use it during the next startup of the listener modules (3).

24. A process according to claim 18, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

25. A process according to claim 14, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

26. A process according to claim 17, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

27. A process according to claim 19, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener

module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

28. A process according to claim 22, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

29. A process according to claim 21, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).

30. A process according to claim 23, characterized in that information related to at least one listener module (3) is displayed and comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run,

the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to a log file of said listener module (LLFPN).--

IN THE ABSTRACT:

Please cancel the Abstract at page 29 in its entirety and substitute the following new Abstract:

Docket 6206
BULL 3587HD

--ABSTRACT

The present invention relates to a process for assisting in the administration of a distributed application of a transaction processing manager based on a binary configuration file (TUXCONFIG), characterized in that said process comprises:

- decompiling the active configuration file of the master machine (Mm),

- retrieving information from the decompiled configuration file of the master machine, and

- checking the consistency of said application running on said given machine.

Information related to at least one listener module is displayed and includes at least the name of the application, the logical name of the machine (LMID) on which the application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of the application, and the access path to a log file of said listener module (LLFPN). If the tlog file containing information on the application running on a given machine does not exist, the file is automatically generated in order to be able to use the file during the next startup of the listener modules.

Docket 6206
BULL 3587HD

REMARKS

This Preliminary Amendment is filed to insert headings to conform the application to U.S. practice, to correct informalities in the specification, claims and abstract resulting from a literal translation of the French text, and to eliminate the use of multiple dependent claims..

Early action on the merits is earnestly solicited.

Respectfully submitted,

KERKAM, STOWELL,
KONDRACKI & CLARKE, P.C.

09380250 .083099

Date: August 30, 1999 BY: [illegible]
Edward J. Kondracki
Registration No. 20,604

Two Skyline Place, Suite 600
5203 Leesburg Pike
Falls Church, VA 22041-3401
Telephone: (703) 998-3302
Telefax: (703) 998-5634

EJK:ah\amdt-pat\BAILLIF-PCT-3587-PREL-AMD

7/PRTS

09/380250
514 Rec'd PCT/PTO 30 AUG 1999

METHOD FOR ASSISTING THE ADMINISTRATION OF A DISTRIBUTED APPLICATION BASED ON A BINARY CONFIGURATION FILE IN A COMPUTER SYSTEM

The present invention relates to a process for assisting in the administration of a distributed application based on a binary configuration file in a computer system. This process for assisting in the administration can especially be applied to a transaction processing manager like the one marketed under the name "Tuxedo."

The "Tuxedo" application allows different software programs that do not recognize one another, but that use a certain protocol, to work together.

Generally, the "Tuxedo" application is a distributed application, i.e., an application that runs on several machines at the same time. A "machine" is the node of the network in which the servers of the "Tuxedo" application run, and the "master machine" is the one that controls the "Tuxedo" application. Fig. 8 illustrates the operation of the "Tuxedo" application. When the "Tuxedo" application is started up, the binary configuration file (TUXCONFIG) is loaded from the disk in the bulletin board (BB) of the master machine (Mm). The bulletin board (BB) represents a set of data structures located in the shared memory and containing information on the transactions, the servers, the services and the clients belonging to the "Tuxedo" application. During the startup of the master machine (Mm), the bulletin board (BB) is loaded into the memory of the master machine (Mm) from a binary "Tuxedo" configuration file (TUXCONFIG). Then, it is distributed to the slave machines (Me) by the master process of the application, called the distinguished bulletin board liaison

(DBBL). Each machine of the application is under the control of a process called a bulletin board liaison (BBL). The distinguished bulletin board liaison DBBL is an administrative process that communicates with the processes (BBL) to coordinate the updates of the bulletin board (BB). The bulletin board liaison BBL is an administrative process that is responsible for maintaining an updated copy of the bulletin board (BB) in its own machine (Me). Each machine (Me) is under the control of a process called BBL, implicitly defined by "Tuxedo." The bridge (BRIDGE) (1) is a process for managing communications between the servers of the "Tuxedo" application. Each machine is provided with a bridge implicitly defined by "Tuxedo." The server TMS (Transaction Manager Server) is a process that manages a validation protocol and recovery for transactions executed by several application servers. The listener module (tlisten, 3) is a process that manages the messages intended for the "Tuxedo" application in a given machine before the bridge process (BRIDGE) of this machine has been started. A listener module allows a machine to receive information coming from other machines. A listener module is required in each machine when the application is distributed.

The "Tuxedo" application is created by the construction of a binary configuration file that defines the architecture of said application (Fig. 7). During the creation of the configuration file, an administrator defines the services (Se) provided by the application and assigns them to application servers (Sr). The administrator then defines groups (G) and assigns a set of servers (Sr). Finally, the administrator assigns groups (G) to a machine (M). Each application must be given a minimum of one group (G), one service (Se) and one server (Sr). A machine (M) can manage several groups (G).

After the creation of a "Tuxedo" application, this application must be administered. The object of the invention is to create a system to assist in the administration of the "Tuxedo" application. The main steps involved in the administration of a "Tuxedo" application consist of:

- a step for loading the binary configuration file of the "Tuxedo" application;
- a step for starting listener modules when the "Tuxedo" application is a distributed application;
- a step for starting the Tuxedo application;
- a step for controlling the application. This consists of displaying information and, if necessary, performing the required corrections;
- a step for stopping the application; and possibly
- a step for stopping the listener modules when they have been started.

The administration of a distributed application can quickly become very complex. In fact, before this administration can begin, the operator must activate a listener module in each slave machine on which he wishes to act. To do this, the administrator must first consult a file containing information on the activation of the listener modules. This file is generally stored, in a place that must be remembered, in each machine. Then, with the aid of this information, the operator must activate the listener module of each machine, one by one. Thus, if the application involves ten machines, the operator must activate the listener module in each of the ten machines, then at the end of the application, deactivate the ten listener modules.

This repetitive operation is long and tedious.

Each administrator has his own solution for performing these tasks. The most common solution is to store in each machine, in a place that must be remembered, scripts for activating the listener modules, and to keep a paper copy of the configuration file. The administrator must make sure that the information is up to date at all times. Each time the configuration changes, he must not forget to print out a paper copy of the configuration file and update the scripts in the slave machines.

Moreover, each time the operator wants to act on an element of an application, he must be able to quickly and accurately identify a given resource, such as for example, when stopping the server "servel" belonging to the group "group1" in the machine "mach1".

When the number of applications increases, these manual operations are the source of numerous errors.

The object of the present invention is to eliminate the drawbacks of the prior art by offering a process for assisting in the administration of a distributed application of a transaction processing manager, based on the binary configuration file of the application, characterized in that said process comprises:

- a step for decompiling the active configuration file of the master machine,
- a step for retrieving information in the decompiled configuration file of the master machine (Mm),
- a step for checking the consistency of said application running on a given machine.

According to another characteristic, said process makes it possible to manage at least one listener module (3) of any machine of the application from another machine.

According to another characteristic, the information related to said distributed application is extracted directly from the active configuration file of the master machine.

According to another characteristic, the step for checking the consistency of said application consists of a comparison between information obtained from the configuration file of the master machine and information obtained from said current application running on another machine.

According to another characteristic, said management of the listener modules makes it possible to start and stop at least one listener module, to display information related to at least one listener module, to change the log of at least one listener module, to check the script of at least one listener module, and to update the script of at least one listener module.

According to another characteristic, an administrator on any machine of the network can start or stop a listener module running on another machine of the network.

According to another characteristic, said process makes it possible to activate several listener modules in a single operation.

According to another characteristic, a graphical interface facilitates the management of the listener modules.

According to another characteristic, said graphical interface makes it possible to display the structure of said application and to select a desired value from a list of values for the current configuration.

According to another characteristic, when the file containing information on said application running on a given machine (tlog) does not exist, the process generates it automatically in order to be able to use it during the next startup of the listener modules (3).

According to another characteristic, said displayed information related to at least one listener module comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the administrator (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to the log file of said listener module (LLFPN).

Other characteristic and advantages of the present invention will emerge more clearly with the reading of the following description given in reference to the attached drawings, in which:

- Fig. 1 represents a window of the graphical interface that offers access to the main commands for managing the modules;

- Fig. 2 represents a window of the graphical interface according to Fig. 1 that makes it possible to activate one or more listener modules;

- Fig. 3 represents a window of the graphical interface according to Fig. 1 that makes it possible to stop one or more listener modules;

- Fig. 4 represents a window of the graphical interface according to claim 1 that makes it possible to display information related to a listener module of a given application;

- Fig. 5 represents a window of the graphical interface according to claim 1 that makes it possible to check the script of a listener module of a given application;

- Fig. 6 represents a window of the graphical interface according to claim 1 that makes it possible to update the script of a listener module in a given machine of a given application;
- Fig. 7 represents the general structure of a distributed application of a transaction processing manager;
- Fig. 8 represents an exemplary application of a transaction processing manager.

The following is a non-limiting exemplary specification of a configuration file. This configuration file, presented in Appendix 1, relates to the "Tuxedo" application. It is divided into six sections (resources, machines, groups, servers, services, network).

The resources section contains general information related to the application. This information is common to all the machines and is constituted by the following parameters:

- IPCKEY, which represents a digital key identifying the shared memory segment in which the application structures are stored. Thanks to this digital key, a given application cannot be in conflict with other applications;
- MASTER, which represents the master machine;
- DOMAINID, which represents the domain of the application;
- MAXACCESSERS, which defines the maximum number of people that can access the application;
- MAXSERVERS, which defines the maximum number of servers that can be connected with the application;
- MAXSERVICES, which defines the maximum number of services that can be connected with the application;
- OPTIONS, which makes it possible to indicate whether the application is running in a local area network;

- MODEL, which makes it possible to indicate whether the application is or is not distributed.

The machines section contains information on each machine (puce, trifide, zig, orage) of the network. This information is constituted by the following parameters:

- LMID (Logical Machine ID), which defines the logical name of the machine, i.e., the name used internally by the application in place of the network name;
- TUXDIR, which specifies the access path to the installation directory of the "Tuxedo" software;
- APPDIR, which specifies the access path to the application servers, i.e., the path leading to the programs of the application (for example, the programs related to the "TUXEDO" application);
- TUXCONFIG, which specifies the absolute access path to the binary configuration file TUXCONFIG, which contains information on the application;
- ENVFILE, which specifies the access path to the file containing the environment variables for the servers and the clients of a given machine;
- ULOGPFX, which specifies the access path to the file "ULOG", which contains information on the history of the application.

The groups section is the section in which each machine is assigned to a group. In the example of Appendix 1, there are four groups. A group is a set of servers that provide related services. In the simplest case, a group is constituted by only one server. All the servers of a group must run on the same machine. An application must comprise at least one group.

The servers section provides information on each server. A server is a module that provides services. In the example of Appendix 1, there are four servers. In the simplest case, a server provides only one service. An application must be provided with at least one server. The server section provides the following information:

- SRVGRP, which defines the group with which the server is affiliated;
- SRVID, which defines the identification number of the server;
- MIN, MAX, which indicates the maximum and minimum occurrences of this server;
- RQADDR, which defines the name of the message queue used for the sending of a message;
- in REPLYQ, the administrator decides on the existence of a response queue;
- CLOPT, which indicates the startup options of the server (available services, priority, etc.).

In the services section, the administrator can specify the services. A service is a set of functions that respond to service requests issued by end users of the application. If the administrator wishes to indicate optional values that are different from the default values, the services must necessarily be defined.

The network section contains, for each machine:

- the complete address used by the bridge process (BRIDGE), called the "Network Address" or "NADDR". The first four digits (0002 in the example of Fig. 4) represent the communication protocol used ("tcp" in the above example). The next four digits represent the port number used by the process and the subsequent digits represent the network address of the machine;

- the access path to the bridge (BRIDGE) of the machine. The bridge is a process for managing communications between the servers of the application. It is used to boot up the application. Each machine is provided with a bridge.

- the complete address of the listener module, called "NLSADDR". The first four digits represent the communication protocol used. The next four digits represent the port number used by the listener module, which must be different from the one used by the bridge process (BRIDGE). The subsequent digits represent the network address of the machine.

The essential characteristic of the invention is that the information related to the application is extracted directly from the active file of the master machine. An administrator on any machine of the network can control the execution of the command "get_tuxval" in the master machine belonging to the administrator, as represented on page [27] of Appendix 2.

The subroutine "get_tuxconfig" of the program used in the implementation of the process for assisting in the administration of a distributed application searches on the hard disk of the master machine for the active configuration file of the application. The latter is then decompiled by means of the command "tmunloadcf" (Page [28] of Appendix 2), lines 85 through 99.

```
get_tuxconfig() {
     if [ -s tuxconf.tmp.$appname ]
     then
          cat tuxconf.tmp.$appname
     else
          rm -f tuxconf.tmp.*
          prog="$Env"'
$TUXDIR/bin/tmunloadcf
echo "\nexit $?"
'
#print -r "$prog" > prog
               rsh "$MASTER" -1 "$ADMIN" "$prog" | tee tuxconf.
tmp.$appname
     fi
get_tlistenlog
}
```

The subroutine "get_tuxval" of this program (Page [28] of Appendix 2, lines 112 through 183) extracts parameters such as LMID, APPDIR, TUXCONFIG, TUXDIR, ROOTDIR, ULOGPFX, NLSADDR, UID and BRIDGE from the binary configuration file of the application obtained by means of the subroutine "get_tuxconfig".

```
get_tuxval() {
     get_tuxconfig | \
     sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/g' | awk '
```

The values of the parameters sought are first initialized. To do this, associative matrices called "tuxconfig_section" are created.

```
BEGIN {
  tuxconfig_section["*RESOURCES"] = 1
  tuxconfig_section["*MACHINES"] =  2
  tuxconfig_section["*GROUPS"] =  3
  tuxconfig_section["*SERVERS"] = 4
  tuxconfig_section["*SERVICES"] = 5
  tuxconfig_section["*ROUTING"] =  6
  tuxconfig_section["*NETWORK"] =  7
```

An index is associated with each matrix. The parameters sought are located in different sections of the configuration file. For example, for the "Tuxedo" application, these different sections, which number seven, are called "Resources," "Machines," "Groups, "Servers," "Services," "Routing" and "Network." In order to be able to extract the parameters that the computer needs, it must be able to mark the place where it is found in the configuration file. In this program, when the field number (NF) is equal to 1, the computer is found at the beginning of a section.

```
NF == 1 {
  if ( $1 in tuxconfig_section ) {
  section = tuxconfig_section[$1]
  next
     }
    }
```

If the computer is in section 2 and the second word is LMID, the computer extracts the logical name of the machine (LMID) on which the administrator is working.

```
section == 2 && $2 == "LMID { # MACHINES section
 if ( $3 == machine) {
   printf "uname=%s\n", $1
   mach_found=1
  }
  else { # reset mach_found for further machines
   mach_found = 0
  }
  next
 }
```

If the computer is in section 2 and the first word is APPDIR, it extracts the access path to the directory under which the servers are bootstrapped.

```
section == 2 && $1 == "APPDIR" && mach_found==1 {
                printf "appdir=%s\n", $2
  appdir = $2
  next
      }
```

Proceeding in the same way, the computer will successively extract, in the machines section of the configuration file, the absolute access path to the binary configuration file (TUXCONFIG), the access path to the installation directory of the Tuxedo software (TUXDIR or ROOTDIR), information on the history of the application (ULOGPFX), and in the network section, the address of the bridge of the machine (NLSADDR).

```
section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
             printf "tuxconfig=%s\n", $2
  next
      }
section == 2 %% $1=="TUXDIR" && mach_found==1{
        printf "tuxdir=%s\n", $2
  next
      }
section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
             printf "tuxdir=%s\n", $2
  next
      }
section == 2 && $1=="ULOGPFX" && mach_found==1 {
  ulogpfx=1; printf "ulogpfx=%s\n", $2
  next
  }
section == 7 && NF == 1 {
 if ( $1 == machine )
   {mach_found = 1}
 else { # reset mach_found for other machines
   mach_found = 0
   }
 next
 }
```

```
section == 7 && $1=="NLSADDR" && mach_found==1 {
       printf "nlsaddr=%s\n", $2
 next
      }
```

The program executes a loop in this subroutine for each machine until the computer finds the current machine. Then, the computer obtains, in the resources section of the configuration file, the identification of the user of the application (UID).

```
section == 1 && $1 == "UID" {printf "uid=%s\n", S2; next }
```

If no value has been defined for the UID in the configuration file, the UID of the person who built the application is used. Next, the computer finds in the network section of the configuration file the access path to the bridge (BRIDGE) of the machine.

```
section == 7 &&     $1=="BRIDGE" && mach_found==1 {
```

The parameter ULOGPFX representing the history of the machine is an optional value. When it does not exist, the computer will generate a file called "ULOG" in the directory APPDIR containing information on the manipulations performed on the application.

```
if ( ulogpfx == 0 ) {
   printf "ulogpfx=%s/ULOG\n", appdir }
        } ' machine=$machine appname=$appname
      lang=`sed -e "s/=/ /g" -e "s/'/ /g" -e "s/;/ /" $ConfDir/
$appname.tux | awk '
       $1 == "LANG" {printf "lang=", $2}'`
   }
```

In addition, the computer needs the working language of the application, represented by the parameter LANG, as well as the value "tlog". The parameter LANG is found in the user's configuration file.

```
lang=`sed -e "s/=/ /g" -e "s/'//g" -e "s/;/ /"
$ConfDir/$appname.tux | awk '
     $1 == "LANG" {printf "lang=", $2}'`
```

The value "tlog" refers to the file "tlistenlog . <name of the application> . <name of the machine>" containing the name of the history file of the listener module.

In the subroutine get_tuxval, the program has gathered all of the environment variables it needs to be able to start the process for assisting in the administration of a distributed application. This process makes it possible, in addition to starting and stopping one or more listener modules, to display information on one or more listener modules, to change the log of one or more listener modules, to check the script of one or more listener modules, and finally, to update the script of one or more listener modules (Fig. 1).

The process for assisting in the administration of a distributed "Tuxedo" application is provided with a graphical interface that allows access to the commands of the transaction processing manager. To execute a task, the administrator is not required to enter commands; he need only click on icons to call up menus and indicate values via dialog boxes. The assisting process is controlled by menus, structured in tree form. The selection of an option in the main menu results in the display of the associated lower level menu. This process is repeated until a pop-up dialog box is displayed, in which the administrator must enter parameter values. In order to be able to manage the listener modules of the distributed "Tuxedo" application, the administrator selects, from the main menu "Tuxedo Commands," the functions "Tuxedo Commands," "Start/Stop Tuxedo Configuration," "Set up a Tuxedo Application" and "Manage the Listener Processes." The selectable functions "Start Listener Processes," "Stop Listener Processes," "Change/Show Listener Process Parameters," "Show currently running Listener Processes," "Check consistency of Listener Process scripts with TUXCONFIG Level" and "Update Listener Process to TUXCONFIG Level" appear in the window of the graphical interface (Fig. 1). To start listener modules, the administrator must select the command "Start Listener Processes" by positioning the cursor of his mouse on the box (11) and pressing on the left button of his mouse. The window of Fig. 2 appears after the selection. If an application has been predesignated, its name is displayed in the box (21). If not, the administrator is informed by the blinking marker of the cursor that he must provide one. To do this, the administrator can either click on the "List" button (23) in order to display the list of the stored applications and select one of them, or explicitly enter the name of the desired application. Next, the administrator is informed by the blinking marker of the cursor in the box (22) that he must indicate the name(s) of the machine(s) in which a listener module must be started. In the same way, the list of the machines comprised in said application can be obtained by clicking on the "List" button (23). In order to validate the machines selected, for example by being highlighted, the administrator must click on the "OK" button (24). The command for starting the listener module is obtained by selecting the "Command" button (25). The "Reset" button (26) makes it possible to reset the values of the boxes (21) and (22). The "?" button (28) offers online help to the administrator.

For each machine designated in the list of machines, the computer obtains information on the application in the configuration file of the master machine, and a history file called "tlistenlog. <name of the application> . <name of the machine>" containing information on the application currently running on this machine. First, the computer checks to see whether the listener module has already been started in the machine. If this is the case, the message "Listener already running on <name of the machine>" is printed on the screen. Otherwise, if a local file exists, the computer executes it and prints the message "Listener started on the machine" if the command succeeds. If the command fails, the computer prints the message "Listener starting failed on <name of the machine>". If the local file does not exist, the computer generates a file "tlistenlog . <name of the application> . <name of the machine>" in the directory APPDIR, executes it, and reports the result as before. This file contains information on the current application and will be used in the next startup of the listener modules. This corresponds to lines 652 through 698 on page [36] and to lines 699 through 719 on page [37] of Appendix 2.

```
startlistproc)
appname=$1; shift
        list="$*"
        set_environ
        loop_status=0
        exit_status=0
        for machine in $list
        do
           echo "\n------- Machine: $machine -------\n"
           get_tuxval > "appname.tux"
      get_tllog
      ../appname.tux
      prog1="
      TUXDIR=$tuxdir; export TUXDIR
         ROOTDIR=$tuxdir; export ROOTDIR # V4
      APPDIR=$appdir; export APPDIR
      TUXCONFIG=$tuxconfig; export TUXCONFIG
      PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
      LANG=$lang; export LANG
      LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
      COLUMNS=200; export COLUMNS
      ps -eF '%u %p %a' | awk'\$3 ~ |"tlisten\" && \$0 ~
\$nlsaddr\" {exit 1}'
      if [ \$? = 1 ]
         then
           echo \"Listener already running on $machine\"
           echo exit 0
             exit 0
                   fi
      if [ -f $appdir/tlisten.$appname.$machine ]
      then
           . $appdir/tlisten.$appname$machine
                ps -eF '%u %p% a' | awk '\$3 ~ \"listen\" && \$0 ~
\$nlsaddr\" {exit 1}'
                if [ \$? = 1 ]
                then
                     echo \"Listener started on $machine\"
                     echo exit 0
                else
                echo \"Listener starting failed on $machine!!!\"
                     echo exit 1
                fi
      else # create the script file & exec it
        echo \"$tuxdir/bin/tlisten -d $bridge -1 $nlsaddr -u $uid
-L $tllog\" > $appdir/tlisten.$appname.$machine
        chmod ug+x $appdir/tlisten.$appname.$machine
        .$appdir/tlisten.$appname.$machine
        ps -eF '%u %p %a'|awk'\$3 ~ \"tlisten\" && \$0 ~
\"nlsaddr\" {exit 1}'
```

```
          if [ \$? = 1 ]
           then
             echo \"Listener started on $machine\"
             echo exit 0
           else
             echo \"Listener starting failed on $machine!!!\"
             echo exit 1
           fi
        fi"
      #echo "$prog1" > prog1
      if [ -z $uname" ]
         then
          [print "Host $machine not found"
          exit 1
      fi
          rsh $uname" -1 $ADMIN" "$prog1" | awk '
             NR == 1 {line = $0}
             NR > 1 ( print line; line = $0 }
             END {if(sub("^exit","", line)) exit line; print line;
exit -1}'
             loop_status=`expr $loop_status\|$?`
             done
              exit $loop _status
        ;;
```

To stop a listener module, the administrator selects, from the main menu for managing listener modules, "Manage the Listener Processes", the function "Stop Listener Processes" by positioning his curser on the box (12) (Fig. 1). The window of Fig. 3 appears. It makes it possible to indicate, in a first box (31), the name of the application, and in a second box (32), the name of the machine or machines. By clicking on the "List" button (33), a list of the applications stored or a list of the machines related to each application can be obtained depending on the position of the blinking position marker (34). For each machine of the application, the computer prints the name of the machine for which the listener module is stopped. This selection on the screen via the graphical interface starts the program steps "stoplistproc" during which the program obtains information, in the station in which the stop procedure is initiated, using get_tuxval on the application contained in the configuration file of the master machine (Page [37] of Appendix 2, lines 720 through 762).

```
stoplistproc)
     appname=$1; shift
     list="$*"
     set_environ
     loop_status=0
     exit_status=0
     for machine in $list
        do
       echo "\n------- Machine: $machine -------\n"
     get_tuxval > "appname.tux"
     ../appname.tux
     prog1="
       COLUMNS=200: export COLUMNS
       ps -eF '%u %p %a'|awk '\$3 ~ \"tlisten\" && $0 ~
\"$nlsaddr\" {print \$2; exit 0} | read pid
       if [ -n\"\$pid\" ]
        then
           kill -9 \$pid > /dev/null
           status=\$?
           if [ \$status -eq 0 ]
           then
               echo \"Process \$pid killed on $machine\"
               echo exit 1
           else
               echo \"Failed to stop listener on $machine!!!\"
               echo exit 1
           fi
        else
           echo \"No Listener running on $machine\"
           echo exit 1
        fi"
     if [ -z "$uname" ]
        then
```

```
            print "Host $machine not found"
            exit 1
        fi
           rsh "$uname" -1 "$ADMIN" "$progl" | awk '
              NR == 1 {line = $0}
              NR > 1 { print line; line = $0 }
              END {if(sub("^exit","", line)) exit line; print line;
exit -1}'
           loop_status=`expr $loop_status \|$?`
        done
      exit $loop_status
    ;;
```

If a process called "tlisten" belonging to the current application is running on this machine, the computer kills it and prints the message "Process <process identifier (PID)> killed on <name of the machine>; otherwise it prints the message "Failed to stop listener on <name of the machine>".

Furthermore, this process for assisting in the administration of an application makes it possible to display information related to a listener module. To do this from the main menu for managing listener modules "Manage the Listener Processes," the administrator need only select the function "Change/Show Listener Processes Parameters" in the box (13) of the window presented in Fig. 1. The window of Fig. 4 appears. The administrator must indicate, in the box (41), the name of the application, and in the box (42), a machine name. As a result of this indication, the other boxes (43 through 46) of the window will show the values of parameters such as:

- the identification of the administrator (UID),
- the complete address of the listener module, composed of the address of the machine and the number of the port it is using (NLSADDR),

- the access path to the network,
- the full access path to the log file of the listener module (Listener Logfile Full Path Name, LLFPN).

All of this information is extracted from the file TUXCONFIG of the master machine. This information cannot be changed by this command, with the exception of LLFPN. Appendix 2 presents, on lines 570 through 579 on page [35], the part of the program corresponding to the execution of the command for changing the LLFPN.

```
chglisten)
     appname=$1
     machine=$2
     shift 2
     if [ $# -gt 0 ]
      then
       echo "TLLOG $machine $1" >
$ConfDir/tlistenlog/$appname.$machine
     fi
     exit $?
          ;
          ;
```

In order to be able to display the active listener modules of the application, the administrator must select the function "Show currently running Listener Processes" by clicking on the box (14) of the window of Fig. 1. The computer displays the list of the machines of the application on which a listener module is active and the process identifier (PID) belonging to the configuration of the network. Appendix 2 presents, on lines 764 through 768 on page [37] and on lines 769 through 809 of page [38], the part of the program corresponding to the display of the list of active listener modules, which uses the function

```
get_tuxval.

running list)
     appname=$1
     loop_status-0
     set_environ
     list_lmids=`get_tuxconfig|\
     sed -e "s/"//g" -e 's/"//g' -e s/\\\\/0/' -e s/\*//" | awk '
       BEGIN { network=0 }
       {line = $0}
          NF == 1 {if (network == 1) print $1}
          $1 == "NETWORK" {network = 1}
          END {if(sub("^exit","",line)) exit line; exit -1 }'`
       for machine in $list_lmids
        do
          get_tuxval > "appname.tux"
          ../appname.tux
          prog1="
          TUXDIR=$tuxdir; export TUXDIR
          LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
          ROOTDIR=$tuxdir; export ROOTDIR # V4
          APPDIR=$appdir; export APPDIR
          TUXCONFIG=$tuxconfig; export TUXCONFIG
          PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
          LANG=$lang; export LANG
          COLUMNS=200; export COLUMNS
          ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~
|"$nlsaddr\" {print \$2}'| read pid
          if [ -n \"\$pid\" ]
          then
             echo \"Listener running on $machine: pid = \$pid\"
             echo exit 0
          else
             echo \"No Listener running on $machine\"
             echo exit 0
          fi"
          if [ -z $uname" ]
             then
               print "Host $machine not found"
               exit 1
          fi
          rsh "$uname" -l "$ADMIN" "$prog1" | awk '
```

```
            NR == 1 {line = $0}
            NR > 1 { print line; line = $0 }
            END { if(sub("^exit","", line)) exit line; print line;
exit -1}'
            loop_status=`expr $loop_status\| $?`
         done
         exit $loop_status
      ;;
```

The administrator can also check the script of a listener module. By selecting the function "Check consistency of Listener Process scripts with Tuxconfig" in the box (15) of the window represented in Fig. 1, the window of Fig. 5 appears. The administrator must enter the name of an application in the box (51) and the name of a given machine in the box (52). A list of the applications and the machines is made available to the administrator by the "List" button (53). The program compares the information contained in the file TUXCONFIG of the master machine and extracted by the function "get_tuxval" with the information contained in the file "tlisten.(name of the application).(name of the machine)" located in the directory APPDIR of the machine and gives the result of this comparison. Appendix 2 presents, on lines 580 through 631 of page [35] and on lines 632 through 651 of page [36], the part of the program corresponding to the checking of a script of a listener module, which makes it possible to indicate the mismatches between the parameters of the files, for example by printing "BRIDGE values mismatch" for the bridge.

```
chklistscript)
                  appname=$1
            machine=$2
            set_environ
            get_tuxval > "appname.tux"
            get_tllog
            ../appname.tux
            prog="
            if [ -f $appdir/tlisten.$appname.$machine ]
             then
              cat $appdir/tlisten.$appname.$machine
              echo \"\\nexit 0\"
            else
              echo \"\\nexit 1\"
            fi"
            if [ -z "$uname" ]
             then
              print "Host $machine not found"
              exit 1
            fi
            rm -f tlscript.$appname$machine
            rsh $uname" -1 "$ADMIN" "$prog" | tee tlscript.
$appname.$machine > /dev/null
            [ $? -ne 0 ] && exit 1
            [ -s tlscript.$appname.$machine ] && cat tlscript.
$appname.$machine|\awk '
            END {if ( $2 == "1" ) exit -1}'
            [ $? -eq -1 ] && exit 1
            [ -s tlscript.$appname.$machine ] && cat tlscript.
$appname.$machine|\
            awk '
            $1 ~ "tlisten" {
              mismatch = 0
              fexec=sprintf("%s/bin/tlisten", tuxdir)
              if ($1 !=fexec){
                 print "tlisten command full pathnames mismatch"
                 printf "\tscript:\t%s\n", $1
                 printf "\tconfig:\t%s\n", fexec
                 mismatch +=1
                 }
              for (i=2; i <= NF; i++) {
                 if (($i == "-d") && ($(i+1) != bridge)){
                   print "BRIDGE values mismatch"
                   printf "\tscript:\t%s\n",$(i+1)
                   printf "\tconfig:\t%s\n", bridge
                   mismatch +=1
                   }
```

```
                    if (( $i == "-1") && ($(i+1) !=nlsaddr)){
                      print "NLSADDR values mismatch"
                      printf "\tscript:\t%s\n", $(i+1)
                      printf "\tconfig:\t%s\n", nlsaddr
                      mismatch +=1
                      }
                    if (($i == "-u") && ($(i+1) != uid)){
                      print "UID values mismatch"
                      printf "\tscript:\t%s\n", $(i+1)
                      printf "\tconfig:\t%s\n", tllog
                      mismatch +=1
                      }
                     }}
                  END {
                    if ( mismatch == 0 )
                     printf "Script File is up-to-date for %s\n",
machine
                    else
                     print f"\nScript File is NOT up-to-date for
%s\n", machine
                    } 'tllog=$tllog machine=$machine bridge=$bridge \
                       nlsaddr=$nlsaddr uid tuxdir=$tuxdir
               exit $?
               ;;
```

A script of a listener module can also be updated by selecting the function "Update Listener Process scripts to TUXCONFIG Level." A script of a Tuxedo listener module makes it possible to start a listener module. It suffices to integrate a script of this type into the startup sequence for a given machine in order for the listening machine to be started automatically at the same time as the machine. In the window represented in Fig. 6, the administrator enters in the box (61) the name of an application, and in the box (62) the name of one or more machines. The program, by calling the subroutine "get_tuxval", obtains all of the information it needs in the binary configuration file extracted by the subroutine "get_tuxconfig" and creates a file corresponding to it in the directory APPDIR under the name "tlisten.(name of the application).(name of the machine). Lines 810 through 831 of Appendix 2, page [38] present the part of the program corresponding to the execution of the command for updating a script of a listener module.

```
updtlistscript)
     appname=$1
     machine=$2
     set_environ
     get_tllog
     get_tuxval > "appname.tux"
     ../appname.tux
     prog="
echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L
$tllog\" > $appdir/tlisten.$appname.$machine
     chmod ug+x $appdir/tlisten.$appname.$machine
     echo exit \$?"
     if [ -z "$uname" ]
      then
        print "Host $machine not found"
        exit 1
     fi
     rsh "$uname" -l "$ADMIN" "$prog" | awk '
      NR == 1 {line = $0}
      NR > 1 { print line; line = $0 }
      END {if(sub("^exit","",line)) exit line; print line; exit
-1}'
     exit $?
     ;;
```

Other modifications within the capability of one skilled in the art are also part of the spirit of the invention.

~~ANNEXE~~ 1 514 Rec'd PCT/PTO 30 AUG 1999 09/380250
APPENDIX

```
#
#       Tuxedo configuration UBBCONFIG for the model TEST1
#

*RESOURCES
IPCKEY                  191785
MASTER                  site1
DOMAINID                        dom1
MAXACCESSERS    50
MAXSERVERS              50
MAXSERVICES             100
OPTIONS                 LAN
MODEL                   MP

*MACHINES
puce            LMID=site1
                TUXDIR="usr/tuxedo"
                APPDIR="/home/dia/tuxedo"
                TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
                ENVFILE="/home/dia/tuxedo/envfile_puce"
                ULOGPFX="/home/dia/tuxedo/ULOG"

trifide         LMID=site2
                TUXDIR="usr/tuxedo"
                APPDIR="/home/dia/tmp"
                TUXCONFIG="/home/dia/tmp/TUXCONFIG"
                ENVFILE="/home/dia/tmp/envfile_trifide"
                ULOGPFX="/home/dia/tmp/ULOG"

zig             LMID=site3
                TUXDIR="usr/tuxedo"
                APPDIR="/home/dia/tuxedo"
                TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
                ENVFILE="/home/dia/tuxedo/envfile_zig"
                ULOGPFX="/home/dia/tuxedo/ULOG"

orage           LMID=site4
                TUXDIR="usr/tuxedo"
                APPDIR="/home/dia/tuxedo"
                TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
                ENVFILE="/home/dia/tuxedo/envfile_orage"
                ULOGPFX="/home/dia/tuxedo/ULOG"



*GROUPS

DEFAULT:        TMSNAME=TMS     TMSCOUNT=2
GROUP1          LMID=site1
                GRPNO=1
GROUP2          LMID=site2
                GRPNO=2
GROUP4          LMID=site3
                GRPNO=3
GROUP3          LMID=site4
                GRPNO=4


*SERVERS
#
DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=Y CLOPT="-A"

SRV1
                SRVGRP=GROUP1
                SRVID=100
                MIN=2   MAX=2
                RQADDR=QSRV1_1
                REPLYQ=Y
                CLOPT="-s SVC1_1 -s SVC1_2 -- "
SRV2
                SRVGRP=GROUP2
```

09380250 " 090500

```
                        SRVID=200
                        MIN=2   MAX=2
                        RQADDR=QSRV2_2
                        REPLYQ=Y
                        CLOPT="-s SVC2_1 -s SVC2_2 -- "
        SRV4
                        SRVGRP=GROUP4
                        SRVID=300
                        MIN=2   MAX=2
                        RQADDR=QSRV4_3
                        REPLYQ=Y
                        CLOPT="-s SVC4_1 -s SVC4_2 -- "
        SRV3
                        SRVGRP=GROUP3
                        SRVID=400
                        MIN=2   MAX=2
                        RQADDR=QSRV3_4
                        REPLYQ=Y
                        CLOPT="-s SVC3_1 -- "


        *SERVICES
        DEFAULT:        LOAD=50
        SVC1_1
        SVC1_2
        SVC2_1
        SVC2_2
        SVC4_1
        SVC4_2
        SVC3_1



        *NETWORK
        site1
        #       port number=60951 (ee17 hexa)
        #       local address=81b683e0
                NADDR="\\x0002ee1781b683e0000000000000000"
                BRIDGE="/dev/xti/tcp"
        #       port number=60952 (ee18 hexa)
                NLSADDR="\\x0002ee1881b683e0000000000000000"
        #
        site2
        #       port number=60951 (ee17 hexa)
        #       local address=81b68387
                NADDR="\\x0002ee1781b68387000000000000000"
                BRIDGE="/dev/xti/tcp"
        #       port number=60952 (ee18 hexa)
                NLSADDR="\\x0002ee1881b68387000000000000000"
        #
        site3
        #       port number=60951 (ee17 hexa)
        #       local address=81b683e1
                NADDR="\\x0002ee1781b683e1000000000000000"
                BRIDGE="/dev/xti/tcp"
        #       port number=60952 (ee18 hexa)
                NLSADDR="\\x0002ee1881b683e1000000000000000"
        #
        site4
        #       port number=60951 (ee17 hexa)
        #       local address=81b6838b
                NADDR="\\x0002ee1781b6838b000000000000000"
                BRIDGE="/dev/xti/tcp"
        #       port number=60952 (ee18 hexa)
                NLSADDR="\\x0002ee1881b6838b000000000000000"
        #

```

```
# @BULL_COPYRIGHT@
#
# HISTORY
# $Log: smtuxadmin.ksh,v $
# Revision 1.7  1996/02/12  11:40:49  odeadm
#       bci V1Set2C 23.01.96
#       [1996/01/23  14:31:07  dia]
#
# Revision 1.6  1995/12/20  14:26:59  odeadm
#       V1 Set2: Still troubles with smtuxadmin.ksh
#       [1995/12/11  11:56:55  odeadm]
#
#       07.12.95 V1Set2 first batch of corrections
#       [1995/12/07  17:22:57  odeadm]
#
#       *** empty log message ***
#       [1995/11/30  13:48:30  dia]
#
#       *** empty log message ***
#       [1995/11/30  13:48:30  dia]
#
# Revision 1.5  1995/10/13  11:52:51  odeadm
#       Servers TMS/Partitioned mach.
#       [1995/10/09  12:05:57  dia]
#
# Revision 1.4  1995/09/15  15:15:06  odeadm
#       Corrections MRs BUILD 3
#       [1995/09/07  15:45:27  dia]
#
# Revision 1.3  1995/08/24  13:38:03  odeadm
#       Build3
#       [1995/08/23  09:04:31  odeadm]
#
# Revision 1.2  1995/07/19  15:18:13  odeadm
#       Madison build M0.2
#       [1995/07/10  10:01:58  odeadm]
#
# $EndLog$
#! /bin/ksh
ConfDir=$WRAPPING_CONFIGURATION
Context=smtuxedo.ctx
Scanconf=$MADISON_VAR/surveyor/scanconf.tux
V5_to_V4='ROOTDIR=$TUXDIR; export ROOTDIR'
Set1_to_Set2='[ -z "$ADMIN" ] && export ADMIN="madison"'
cmd=$1; shift

set_environ() {
        MASTER=""; APPDIR=""; ADMIN=""
        filename=$ConfDir/$appname.tux
Env=`tuxgetenv -k -v APP_PW $filename << !
tuxgetenvp
!`
        eval "$Env"; unset APP_PW
        eval "$Set1_to_Set2"
        if [ -n "$MASTER" -a -n "$APPDIR" ]
        then
                Env="$Env
$PW
$Set1_to_Set2
$V5_to_V4"'
LD_LIBRARY_PATH=$LIBPATH; export LD_LIBRARY_PATH;
cd $APPDIR
PATH=${PATH}:.:$APPDIR:$TUXDIR/bin; export PATH'
                return 0
        fi
        exit 1
}

remote_cmd() {
        prog="$Env
```

```
$cmd"'
status=$?
sleep 1
echo "\nexit $status"
'
#print -r "$prog" > prog
        rsh "$MASTER" -l "$ADMIN" "$prog" | awk '
                NR == 1 {line = $0}
    NR > 1  { print line; line = $0}
                END {if(sub("^exit ","", line)) exit line; exit -1 }'
}


get_tuxconfig() {
        if [ -s tuxconf.tmp.$appname ]
        then
                cat tuxconf.tmp.$appname
        else
                rm  -f tuxconf.tmp.*
                prog="$Env"'
$TUXDIR/bin/tmunloadcf
echo "\nexit $?"
'
#print -r "$prog" > prog
                rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
        fi
get_tlistenlog
}

get_tlistenlog() {
        tllogfname=$ConfDir/tlistenlog.$appname.$machine
if [ -s $tllogfname ]
 then
   cat $tllogfname
 else # default value
   echo "TLLOG $machine $MADISON_TMP/tlisten.$appname.$machine.log" | tee $tllogfname
fi
   echo "\nexit $?"
}

get_tuxval() {
        get_tuxconfig | \
        sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/g' | awk '
  BEGIN {
    tuxconfig_section["*RESOURCES"] = 1
    tuxconfig_section["*MACHINES"]  = 2
    tuxconfig_section["*GROUPS"]  = 3
    tuxconfig_section["*SERVERS"] = 4
    tuxconfig_section["*SERVICES"]  = 5
    tuxconfig_section["*ROUTING"] = 6
    tuxconfig_section["*NETWORK"] = 7
}
 NF == 1 {
    if ( $1 in tuxconfig_section ) {
    section = tuxconfig_section[$1]
    next
    }
  }
 section == 2 && $2 == "LMID" {  # MACHINES section
 if ( $3 == machine) {
   printf "uname=%s\n", $1
   mach_found=1
  }
  else { # reset mach_found for furtheur machines
   mach_found = 0
  }
  next
 }
section == 2 && $1=="APPDIR" && mach_found==1 {
               printf "appdir=%s\n", $2
    appdir = $2
    next
       }
section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
             printf "tuxconfig=%s\n", $2
    next
       }
section == 2 && $1=="TUXDIR" && mach_found==1 {
       printf "tuxdir=%s\n", $2
  next
       }
section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
             printf "tuxdir=%s\n", $2
    next
       }
section == 2 && $1=="ULOGPFX" && mach_found==1 {
    ulogpfx=1; printf "ulogpfx=%s\n", $2
    next
    }
section == 7 && NF == 1 {
  if ( $1 == machine )
    {mach_found = 1}
  else { # reset mach_found for other machines
    mach_found = 0
    }
  next
  }
section == 7 && $1=="NLSADDR" && mach_found==1 {
       printf "nlsaddr=%s\n", $2
  next
       }
section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
section == 7 &&     $1=="BRIDGE" && mach_found==1 {
             printf "bridge=%s\n", $2 }
  END { # not defined ulogpfx
   if ( ulogpfx == 0 ) {
    printf "ulogpfx=%s/ULOG\n", appdir }
       } ' machine=$machine appname=$appname
      lang=`sed -e "s/=/ /g" -e "s/'//g" -e "s/;/ /" $ConfDir/$appname.tux | awk '
        $1 == "LANG" {printf "lang=", $2}' `
    }

get_tllog() {
tllogfname="$ConfDir/tlistenlog.$appname.$machine"
if [ -f $tllogfname ]
 then
  tllog=`cat $tllogfname|awk '$1 == "TLLOG" && $2 == machine { print $3 }' machine=$m
achine`
 else
   tllog="$MADISON_TMP/tlistenlog.$appname.$machine"
   echo "TLLOG $machine $tllog" > $tllogfname
fi
}


case $cmd in
      appli)
            ls -l $ConfDir 2> /dev/null | awk '
                  sub(".tux$", "", $NF) {print $NF}'
            ;;
      isexist)
            if [ -f $ConfDir/$1.tux ]
            then
                  echo "Yes"
            else
                  echo "No"
            fi
            ;;
      setparam)
            [ ! -d $ConfDir ] && mkdir -p $ConfDir
            if [ -n "$2" ]
            then
```

```
                        filename=$ConfDir/$2.tux
                        while [ $# -gt 0 ]
                        do
                                echo "$1=\"$2\"; export $1"
                                shift 2
                        done > $filename
                fi
                ;;
        discover)
                [ -z "$1" ] && exit 1
                filename=$ConfDir/$1.tux; shift
                if [ -f $filename ]
                then
        #               sed -e 's/:/@@@/g' -e 's/#.*//' -e 's/ *; */"/g' $filename/ | awk '
                        sed  -e 's/#.*//' -e 's/ *; */"/g' -e 's/:/#!:/g' $filename/ | awk '
                        BEGIN { field = "#promptW:promptP:promptPO:promptS:promptA:promptM:promptC:promptR:promptF"; value=":::::::" }
                        /\=/ {
                                for (i=1; i<= NF; i++) {
                                     if(sub("=$", "", $i)) {
                                        separator = ":"
                                        field = field separator $i
                                        value = value separator $(i+1)
                                     }
                                }
                        }
                        END {
                                print field; print value
                        }' FS='"'
                else
                        print '#\n'
                fi
                ;;
        delappname)
                if [ -n "$2" ]
                then
                        filename=$ConfDir/$2.tux
                        if [ -f $filename ] && grep -q "$1=['\"]*$2" $filename
                        then
                                rm -f $filename ${filename}p
                        else
                                echo 'The file does not exist'
                                echo '           or'
                                echo 'The file is not an environment file'
                                exit 1
                        fi
                fi
                ;;
        select)
                if [ -n "$2" ]
                then
                        echo "$1='$2'; export $1" > "$Context"
                fi
                ;;
        deselect)
                rm -f "$Context"
                ;;
        selected)
                APPNAME=""
                [ -f $Context ] && . ./$Context
                echo "$1$APPNAME"
                ;;
        isselected)
                rm -f tuxconf.tmp.*
                [ -f $Context ] && fgrep -q "APPNAME=" $Context && shift
                echo $1
                ;;
        loadcf)
                appname=$1
```

loop

```
    boucle_status=0
             cmd="\$TUXDIR/bin/tmloadcf -y $2 $3"
             set_environ
   echo "---- Loading Configuration Binary File ---"
             remote_cmd
    status=$?
    if [ $status -ne 0 ]
     then
      exit $status
     else
# maj fichier $Scanconf.tux machines
       prog="$Env"'
       $TUXDIR/bin/tmunloadcf
       echo "\nexit $?"
       '
       #print -r "$prog" > prog
       rsh "$MASTER" -l "$ADMIN" "$prog" > tuxconf.tmp.$appname
       list_lmids=`cat tuxconf.tmp.$appname | sed -e "s/=/ /g" -e 's/"//g' -e "s/\*//
"  | awk '
        {line = $0}
        $2 == "LMID" && machine == 1 {lmids = lmids $3 " "; next}
        $1 == "GROUPS" && $2 == "" { machine=0; next}
        $1 == "MACHINES" && $2 == "" { machine = 1; next}
        END {if(sub("^exit ","", line)) {
           print lmids
           exit line}
           exit -1 }'`
       for machine in $list_lmids
         do
         echo "---- Updating $Scanconf on $machine ----\n"
         get_tuxval > "appname.tux"
         . ./appname.tux
         log_prefix=`echo $ulogpfx | sed -e 's./. .g' | awk '
             {print $NF} '`
         log_dir=`echo $ulogpfx | sed -e 's./. .g' | awk '
           {for (i=1; i< NF; i++) {
             tempo = tempo "/" $i }}
           END { print tempo}'`
#Build the 3 lines of $Scanconf for the application
         prog="
[ -x $MADISON_BIN/security/updscantux ] &&
$MADISON_BIN/security/updscantux $appname $log_dir $log_prefix
echo \"\\nexit \$?\""
          rsh "$uname" -l madison "$prog"  | awk '
             NR == 1 {line = $0}
             NR > 1  { print line; line = $0}
             END {if(sub("^exit ","", line)) exit line; exit -1 }'
        boucle_status=`expr $boucle_status + $? `
      done
     fi
     exit $boucle_status
             ;;
      apppwd)
             filename=$ConfDir/$1.tuxp
             echo "Enter Application Password: \c"
             OLDCONFIG=`stty -g`
             stty -echo
             read APP_PW
             echo "\nRe-enter Application Password: \c"
             read APP_PW_1
             stty $OLDCONFIG
             if [ "$APP_PW" != "$APP_PW_1" ]
             then
                     echo "\n\nPassword mismatch!"
                     echo "Enter any character to exit and retry"
                     read
             else
     #               PWencode "APP_PW=\"$APP_PW\"; export APP_PW" > $filename
       #                 APP_PW=`echo $APP_PW | sed -e "s/'/'\"'\"'/g"`
     #               PWencode "APP_PW='$APP_PW'; export APP_PW" > $filename
tuxgetenv -s > $filename << !
```

```
tuxgetenvp
$APP_PW
!
                fi
                ;;
        chksyntx)
                appname=$1
                cmd="\$TUXDIR/bin/tmloadcf -n $2"
                set_environ
                remote_cmd
                exit $?
                ;;
        dispIpc)
                appname=$1
                cmd="\$TUXDIR/bin/tmloadcf -c $2"
                set_environ
                remote_cmd
                exit $?
        ;;
        machine_network)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { network=0 }
                        {line = $0}
                        NF == 1 { if (network == 1) print $1}
                        $1 == "NETWORK" { network = 1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;

        machine_machines)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { machine=0 }
                        {line = $0}
                        $2 == "LMID" { if(machine == 1) print $3}
                        $1 == "GROUPS" { if( $2 == "" ) machine=0}
                        $1 == "MACHINES" { if( $2 == "" ) machine = 1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;
        group)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { group=0 }
                        {line = $0}
                        $1 == "SERVERS" { group=0 }
                        $1 == "GROUPS"  { if($2 == "") group=1}
                        $2 == "LMID" && $4 == "GRPNO" { if(group) print $1}
                        END {if(sub("^exit ","", line)) exit line; exit -1 }'
                exit $?
                ;;
        svrname)
                appname=$1
                set_environ
                get_tuxconfig | \
                sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                        BEGIN { group=server=nb_of_distinct_svr_name=0 }
                        {line = $0}
                        $1 == "TMSNAME" { if ( group == 1) {
                            trouve = 0
                            if (nb_of_distinct_svr_name == 0) {
                               nb_of_distinct_svr_name=1
                               svr_names[nb_of_distinct_svr_name] = $2
                               print $2
                ) else {
                  for (j=1; j<= nb_of_distinct_svr_name; j++) {
                        if ( $2 == svr_names[j] ) {
                              trouve=1
                        }
                  }
                  if (trouve == 0) {
                        nb_of_distinct_svr_name += 1
                        svr_names[nb_of_distinct_svr_name] = $2
                        print $2
                      }
                  }
                }
            }
            $1 == "SERVERS" { if ($2 == "") {
                  server=1
                  group=0 }
            }
            $1 == "SERVICES" { if ($2== "") server=0}
            $1 == "GROUPS"   { if ($2 == "") group=1}
            $2 == "SRVGRP" {
                if((server == 1) && ( $4 == "SRVID")) {
                  trouve = 0
                  if (nb_of_distinct_svr_name == 0) {
                      nb_of_distinct_svr_name = 1
                      svr_names[nb_of_distinct_svr_name] = $1
                      print $1
                  } else {
                      for(j=1; j<= nb_of_distinct_svr_name; j++) {
                        if ( $1 == svr_names[j] ) {
                              trouve=1
                        }
                      }
                      if(trouve == 0) {
                        nb_of_distinct_svr_name += 1
                        svr_names[nb_of_distinct_svr_name] = $1
                        print $1
                      }
                  }
                }
            }
            END {if(sub("^exit ","", line)) exit line; exit -1 }'
      exit $?
      ;;
svrseq)
      appname=$1
      set_environ
      get_tuxconfig | \
      sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
            BEGIN { server=0; nb_of_distinct_svr_seq=0 }
            {line = $0}
            $1 == "SEQUENCE" && server == 1 {
                trouve = 0
                if (nb_of_distinct_svr_seq == 0) {
                  nb_of_distinct_svr_seq=1
                  svr_seqs[nb_of_distinct_svr_seq] = $2
                  print $2
                } else {
                  for (j=1; j<= nb_of_distinct_svr_seq; j++) {
                        if ( $2 == svr_seqs[j] ) {
                              trouve=1
                        }
                  }
                  if (trouve == 0) {
                        nb_of_distinct_svr_seq += 1
                        svr_seqs[nb_of_distinct_svr_seq] = $2
                        print $2
                  }
                }
            }
            $1 == "SERVERS" {  if($2 == "") server=1}
```

```
                $1 == "SERVICES" { if($2 == "") server=0}
                END {if(sub("^exit ","", line)) exit line; exit -1 }'
            exit $?
            ;;
    svrId)
            appname=$1
            set_environ
            get_tuxconfig | \
            sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
                BEGIN { server=0; nb_of_distinct_svr_Id=0 }
                {line = $0}
                $2 == "SRVGRP" && $4 == "SRVID" &&  server == 1 {
                    trouve = 0
                    if (nb_of_distinct_svr_Id == 0) {
                      nb_of_distinct_svr_Id=1
                      svr_Ids[nb_of_distinct_svr_Id] = $5
                      print $5
                    } else {
                      for (j=1; j<= nb_of_distinct_svr_Id; j++) {
                          if ( $5 == svr_Ids[j] ) {
                              trouve=1
                          }
                      }
                      if (trouve == 0) {
                          nb_of_distinct_svr_Id += 1
                          svr_Ids[nb_of_distinct_svr_Id] = $5
                          print $5
                      }
                    }
                }
                $1 == "SERVERS" {  if($2 == "") server=1}
                $1 == "SERVICES" { if($2 == "") server=0}
                END {if(sub("^exit ","", line)) exit line; exit -1 }'
            exit $?
            ;;
    discover_conf)
            machine=$2
            appname=$1
            set_environ
            get_tuxconfig | \
            sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/' -e "s/\*//"  | awk '
                BEGIN {field = "#" }
                {line = $0}
                $1 == "UID" {
                      field = field separator $1
                      value = value separator $2
                      separator = ":"
                }
                $1 == "GID" {
                      field = field separator $1
                      value = value separator $2
                      separator = ":"
                }

                $1 == "BRIDGE" && network == 1 && mach_found == 1 {
                      field = field separator $1
                      value = value separator $2
                }
                   $1 == "NLSADDR" && network == 1 && mach_found == 1 {
                         field = field separator $1
                         value = value separator $2
                      network = 0
                      mach_found = 0
                   }
                   $1 == "TLLOG" && $2 == machine {
                         field = field separator $1
                      value = value separator $3
                   }

                $1 == machine {mach_found = 1}
                $1 == "NETWORK"  { network = 1}
            END {
                print field; print value
                if(sub("^exit ","", line)) exit line; exit -1
            }' "machine=$machine"
        exit $?
        ;;
    chglisten)
          appname=$1
          machine=$2
          shift 2
          if [ $# -gt 0 ]
           then
            echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
          fi
          exit $?
        ;;
    chklistscript)
        appname=$1
          machine=$2
          set_environ
          get_tuxval > "appname.tux"
          get_tllog
          . ./appname.tux
          prog="
          if [ -f $appdir/tlisten.$appname.$machine ]
           then
            cat $appdir/tlisten.$appname.$machine
            echo \"\\nexit 0\"
           else
            echo \"\\nexit 1\"
          fi"
          if [ -z "$uname" ]
           then
            print "Host $machine not found"
            exit 1
          fi
          rm -f tlscript.$appname.$machine
          rsh "$uname" -l "$ADMIN" "$prog" | tee tlscript.$appname.$machine > /dev/null
          [ $? -ne 0 ] && exit 1
          [ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine | awk '
           END { if ( $2 == "1" ) exit -1} '
          [ $? -eq -1 ] && exit 1
          [ -s tlscript.$appname.$machine ] && cat tlscript.$appname.$machine | \
          awk '
           $1 ~ "tlisten" {
             mismatch = 0
             fexec=sprintf("%s/bin/tlisten", tuxdir)
             if ($1 != fexec) {
               print "tlisten command full pathnames mismatch"
               printf "\tscript:\t%s\n", $1
               printf "\tconfig:\t%s\n", fexec
               mismatch +=1
               }
             for (i=2; i <= NF; i++) {
               if (( $i == "-d") && ($(i+1) != bridge)) {
                 print "BRIDGE values mismatch"
                 printf "\tscript:\t%s\n",$(i+1)
                 printf "\tconfig:\t%s\n",bridge
                 mismatch +=1
                 }
                if (( $i == "-l") && ($(i+1) !=nlsaddr)) {
                 print "NLSADDR values mismatch"
                 printf "\tscript:\t%s\n",$(i+1)
                 printf "\tconfig:\t%s\n", nlsaddr
                 mismatch +=1
                 }
                if (( $i == "-u") && ($(i+1) != uid)) {
                 print "UID values mismatch"
```

```
                           printf "\tscript:\t%s\n", $(i+1)
                           printf "\tconfig:\t%s\n",uid
                           mismatch +=1
                           }
                         if (( $i == "-L") && ($(i+1) !=tllog)) {
                           print "LOGFILE values mismatch"
                           printf "\tscript:\t%s\n", $(i+1)
                           printf "\tconfig:\t%s\n", tllog
                           mismatch +=1
                           }
                         }}
                  END {
                     if ( mismatch == 0 )
                      printf "Script File is up-to-date for %s\n",machine
                     else
                      printf "\nScript File is NOT up-to-date for %s\n",machine
                     } ' tllog=$tllog machine=$machine bridge=$bridge \
                          nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
                exit $?
                ;;
      startlistproc)
             appname=$1; shift
             list="$*"
             set_environ
             boucle_status=0                        (boucle = loop)
             exit_status=0
             for machine in $list
             do
                echo "\n------- Machine: $machine -------\n"
                get_tuxval > "appname.tux"
                   get_tllog
                   . ./appname.tux
                   prog1="
                   TUXDIR=$tuxdir; export TUXDIR
                ROOTDIR=$tuxdir; export ROOTDIR  # V4
                   APPDIR=$appdir; export APPDIR
                   TUXCONFIG=$tuxconfig; export TUXCONFIG
                   PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
                   LANG=$lang; export LANG
                   LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
                   COLUMNS=200; export COLUMNS
                   ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {
exit 1}'
                   if [ \$? = 1 ]
                     then
                        echo \"Listener already running on $machine\"
                        echo exit 0
                      exit 0
                            fi
                   if [ -f $appdir/tlisten.$appname.$machine ]
                   then
                        . $appdir/tlisten.$appname.$machine
                      ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nls
addr\" {exit 1}'
                      if [ \$? = 1 ]
                      then
                         echo \"Listener started on $machine\"
                         echo exit 0
                      else
                        echo \"Listener starting failed on $machine !!!\"
                         echo exit 1
                      fi
                    else # create the script file & exec it
                          echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L
$tllog\" > $appdir/tlisten.$appname.$machine
                        chmod ug+x $appdir/tlisten.$appname.$machine
                         . $appdir/tlisten.$appname.$machine
                        ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsadd
r\" {exit 1}'
                        if [ \$? = 1 ]
                         then
```

```
                              echo \"Listener started on $machine\"
                              echo exit 0
                            else
                                echo \"Listener starting failed on $machine !!!\"
                                echo exit 1
                            fi
                      fi"
          #echo "$progl" > progl
          if [ -z "$uname" ]
            then
               print "Host $machine not found"
               exit 1
          fi
        rsh "$uname" -l "$ADMIN" "$progl" | awk '
             NR == 1 {line = $0}
             NR > 1 { print line; line = $0 }
             END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
        boucle_status=`expr $boucle_status \| $? `
             done
            exit $boucle_status
      ;;
    stoplistproc)
        appname=$1; shift
        list="$*"
        set_environ
        boucle_status=0
        exit_status=0
        for machine in $list
          do
           echo "\n------- Machine: $machine -------\n"
            get_tuxval > "appname.tux"
            . ./appname.tux
            progl="
           COLUMNS=200; export COLUMNS
           ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print \$
2; exit 0 }' | read pid
           if [ -n \"\$pid\" ]
             then
               kill -9 \$pid > /dev/null
               status=\$?
               if [ \$status -eq 0 ]
               then
                       echo \"Process \$pid killed on $machine\"
                       echo exit 0
               else
                       echo \"Failed to stop listener on $machine!!!\"
                       echo exit 1
               fi
           else
               echo \"No Listener running on $machine\"
               echo exit 1
           fi"
            if [ -z "$uname" ]
               then
                  print "Host $machine not found"
                  exit 1
            fi
           rsh "$uname" -l "$ADMIN" "$progl" | awk '
               NR == 1 {line = $0}
               NR > 1 { print line; line = $0 }
               END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
           boucle_status=`expr $boucle_status \| $? `
            done
        exit $boucle_status
      ;;

          runninglist)
           appname=$1
           boucle_status=0
           set_environ
           list_lmids=`get_tuxconfig | \
    sed -e "s/=/ /g" -e 's/"//g' -e 's/\\\\/0/' -e "s/\*//"  | awk '
       BEGIN { network=0 }
       {line = $0}
       NF == 1 { if (network == 1) print $1}
       $1 == "NETWORK" { network = 1}
       END {if(sub("^exit ","", line)) exit line; exit -1 }' `
    for machine in $list_lmids
      do
        get_tuxval > "appname.tux"
        . ./appname.tux
        prog1="
        TUXDIR=$tuxdir; export TUXDIR
        LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
        ROOTDIR=$tuxdir; export ROOTDIR  # V4
        APPDIR=$appdir; export APPDIR
        TUXCONFIG=$tuxconfig; export TUXCONFIG
        PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
        LANG=$lang; export LANG
        COLUMNS=200; export COLUMNS
        ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"$nlsaddr\" {print \$2}' | read pid
        if [ -n \"\$pid\" ]
         then
           echo \"Listener running on $machine: pid = \$pid\"
           echo exit 0
         else
           echo \"No Listener running on $machine\"
           echo exit 0
        fi"
        if [ -z "$uname" ]
          then
            print "Host $machine not found"
            exit 1
        fi
        rsh "$uname" -l "$ADMIN" "$prog1" | awk '
         NR == 1 {line = $0}
         NR > 1  { print line; line = $0}
         END { if (sub("^exit ", "", line)) exit line; print line; exit -1} '
         boucle_status=`expr $boucle_status \| $?`
      done
      exit $boucle_status
   ;;
   updtlistscript)
   appname=$1
   machine=$2
   set_environ
   get_tllog
   get_tuxval > "appname.tux"
   . ./appname.tux
   prog="
   echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" >  $appdir/tlisten.$appname.$machine
   chmod ug+x $appdir/tlisten.$appname.$machine
   echo exit \$?"
   if [ -z "$uname" ]
     then
       print "Host $machine not found"
       exit 1
   fi
   rsh "$uname" -l "$ADMIN" "$prog" | awk '
     NR == 1 {line = $0}
     NR > 1 { print line; line = $0 }
     END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
   exit $?
   ;;
  tuxBootEnt)
        appname=$1; shift
        cmd="\$TUXDIR/bin/tmboot -y $@"
        set_environ
        remote_cmd
        exit $?
```

```
                ;;
        tuxShutEnt)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmshutdown -y"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxBootAllMach)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmboot -y -A  $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxShutAllMach)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmshutdown -y -A $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxShut)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmshutdown -y $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxShutAdmMast)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmshutdown -y -M $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxShutSvrSect)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmshutdown -y -S $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxBootAdmMast)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmboot -y -M $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxBoot)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmboot -y $@"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxShutdown)
                appname=$2
                cmd="\$TUXDIR/bin/tmshutdown -y $1"
                set_environ
                remote_cmd
                exit $?
                ;;
        tuxBootSvrSct)
                appname=$1; shift
                cmd="\$TUXDIR/bin/tmboot -y -S $@"
                set_environ
                remote_cmd
                exit $?
                ;;
```

```
    tuxBootBBL)
           #echo $*
        appname=$1; shift
        cmd="\$TUXDIR/bin/tmboot -y $@"
        set_environ
        remote_cmd
        exit $?
        ;;
    tuxShowBooted)
        appname=$1; shift
        cmd="(echo psr; echo quit)|\$TUXDIR/bin/tmadmin"
        set_environ
        remote_cmd
        exit $?
        ;;
    tuxminIPC)
        appname=$1; shift
        cmd="\$TUXDIR/bin/tmboot -y -c $@"
        set_environ
        remote_cmd
        exit $?
        ;;
    tuxShutPart)
           exit_status=0
        appname=$1;
        machine=$2; shift
        set_environ
        get_tuxconfig | \
        sed -e "s/=/ /g" -e 's/"//g' -e 's/\\//' -e "s/\*//"  | awk '
            $1 == "APPDIR" && mach_section == 1 && mach_found == 1 {
                print "APPDIR " $2 > "appname.tux"
                mach_section = 0
                mach_found = 0
            }
            $1 == "TUXCONFIG" && mach_section==1 && mach_found==1 {
                print "TUXCONFIG " $2 > "appname.tux"
            }
            $1 == "MACHINES" {mach_section = 1}
            $2 == "LMID" && mach_section == 1 && $3 == machine {
                print "MACHINE " $1 > "appname.tux"
                mach_found = 1
            }
            $1 == "TUXDIR" && mach_section==1 && mach_found==1 {
                print "TUXDIR " $2 > "appname.tux"
            }
        ' "machine=$machine" "appname=$appname"
        if [ $? != 0 ]
        then
            exit 1
        fi
       appdir=`awk '$1 == "APPDIR" {print $2}' appname.tux`
       tuxconfig=`awk '$1 == "TUXCONFIG" {print $2}' appname.tux`
       uname=`awk '$1 == "MACHINE" {print $2}' appname.tux`
       rootdir=`awk '$1 == "TUXDIR" {print $2}' appname.tux`
       lang=`sed -e 's/=/ /g' -e 's/;/ /g' $ConfDir/$appname.tux |
            awk '$1 == "LANG" {print $2}'`
       prog1="TUXDIR=$rootdir; export TUXDIR
             APPDIR=$appdir; export APPDIR
             LIBPATH=${LIBPATH}:$rootdir/lib; export LIBPATH
             TUXCONFIG=$tuxconfig; export TUXCONFIG
             LANG=$lang; export LANG
             PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
             $TUXDIR/bin/tmshutdown -y -P $@
             echo \$? > /tmp/rem$appname.$machine.tux"
       if [ -z "$uname" ]
         then
           print "Host $machine not found"
           exit 1
       fi
       rsh $uname -l "$ADMIN" "$prog1"
       rsh_status=`echo $?`
```

```
            if [ "$rsh_status" -eq "0" ]
              then
               status=`rsh $uname -l "$ADMIN" "cat /tmp/rem$appname.$machine.tux"`
               rsh $MASTER -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/null
               rsh $uname -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux"  2> /dev/null
            fi
            if [ "$status" -ne "0" ]
              then
                 exit_status=`expr $exit_status + 1`
            fi
            if [ "$exit_status" -ne "0" -o "$rsh_status" -ne "0" ]
              then
                exit 1
            fi
    ;;
    loadfshm)
    appname=$1; machine=$2; shift 2
    set_environ
    get_tuxval > "appname.tux"
    . ./appname.tux
    prog="
    TUXDIR=$tuxdir; export TUXDIR
    ROOTDIR=$tuxdir; export ROOTDIR
    LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
    LANG=$lang; export LANG
    $tuxdir/bin/loadfiles $@
    echo \"\nexit \$?\""
    if [ -z "$uname" ]
      then
        print "Host $machine not found"
        exit 1
    fi
    rsh "$uname" -l "$ADMIN" "$prog" | awk '
       NR == 1 {line = $0}
       NR > 1 { print line; line = $0 }
       END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
    ;;
    Unloadcf)
    appname=$1
    set_environ
    cmd="\$TUXDIR/bin/tmunloadcf"
    if [ $# -eq 2 ]
      then
       filename=$2
       remote_cmd > "$filename"
    else
       remote_cmd
    fi
    exit $?
    ;;
   *)
    echo "Command $1 does not exist"
    exit 1
    ;;
esac
```

CLAIMS

1. Process for assisting in the administration of a distributed application of a transaction processing manager, based on a binary configuration file (TUXCONFIG), characterized in that said process comprises:

- a step for retrieving information related to said application in a configuration file of a master machine (Mm),
- a step for checking the consistency of said application running on a given machine.

2. Process according to claim 1, characterized in that it comprises a step for managing at least one listener module (3) of any machine of the application from another machine.

3. Process according to claim 1, characterized in that the information related to said distributed application is extracted directly from the active configuration file of the master machine.

4. Process according to claim 1, characterized in that the step for checking the consistency of said application consists of a comparison between the information obtained from the configuration file of the master machine and the information obtained from said current application running on a given machine.

5. Process according to claim 2, characterized in that said administration of the listener modules consists of starting

and stopping at least one listener module, displaying information related to at least one listener module, changing the log of at least one listener module, checking the script of at least one listener module and/or updating the script of at least one listener module.

6. Process according to claim 2, characterized in that it comprises a step for starting and stopping a listener module running on a first machine, this step being carried out by an administrator using a second machine distinct from the first one, belonging to the same network as the first machine.

7. Process according to claim 2, characterized in that it comprises a step for simultaneously activating several listener modules.

8. Process according to claim 1, characterized in that it comprises a step for decompiling the active configuration file of the master machine.

9. Process according to claim 2, characterized in that the steps of the process are implemented by means of a graphical interface comprising at least one icon, at least one menu and at least one dialog box.

10. Process according to claim 9, characterized in that the menus of the graphical interface are structured in tree form and the activation of a menu results in the display of a list of values of the current configuration, selectable by the user.

11. Process according to claim 4, characterized in that when the file containing information on said application running on a given machine (tlog) does not exist, the process generates it automatically in order to be able use it during the next startup of the listener modules (3).

12. Process according to claim 6, characterized in that said displayed information related to at least one listener module (3) comprises at least the name of said application, the logical name of the machine (LMID) on which said application is run, the identification of the user (UID) of said application, the address used by the listener module (NLSADDR), the access path to the network of said application, and the access path to the log file of said listener module (LLFPN).

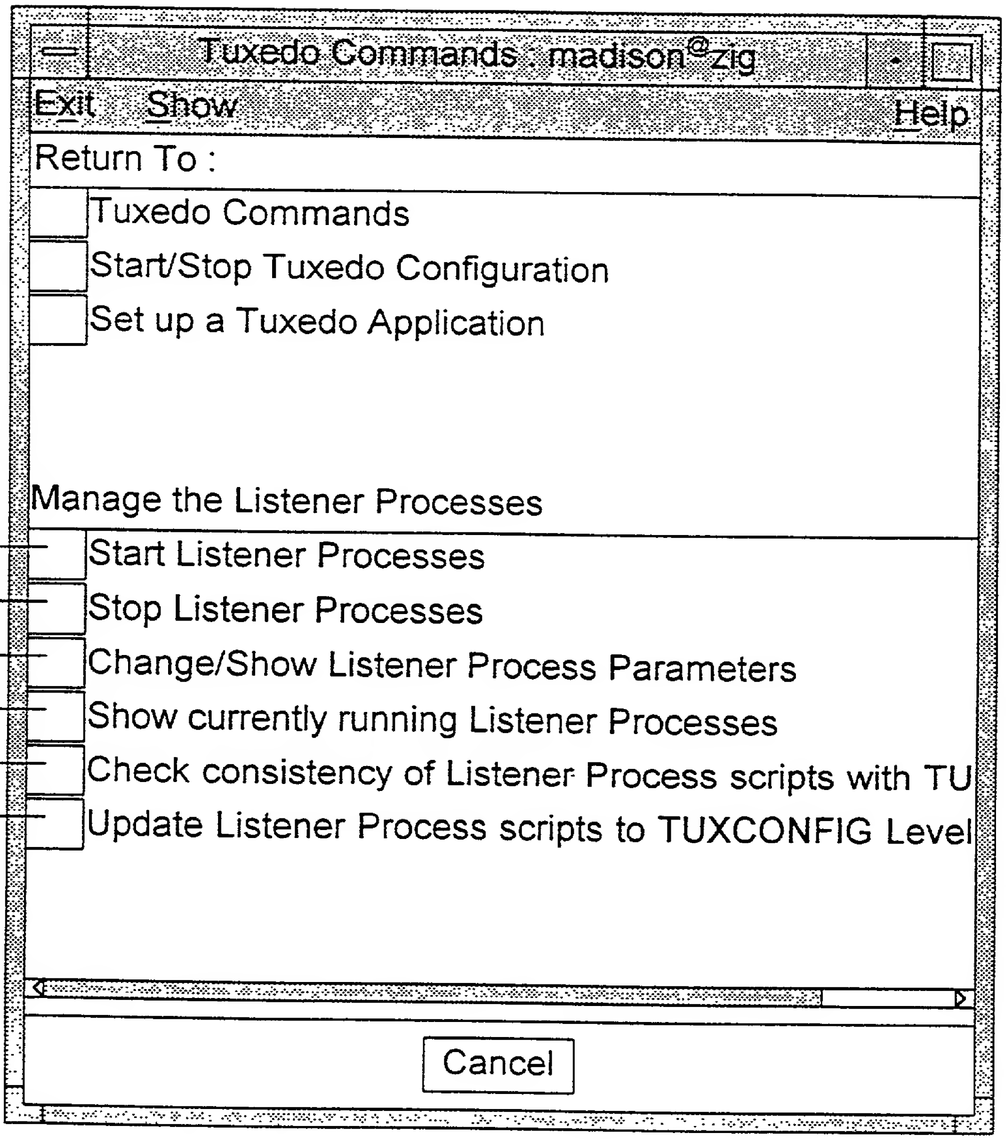
ABSTRACT

The present invention relates to a process for assisting in the administration of a distributed application of a transaction processing manager based on a binary configuration file (TUXCONFIG), characterized in that said process comprises:

- a step for decompiling the active configuration file of the master machine (Mm),
- a step for retrieving information from the decompiled configuration file of the master machine,
- a step for checking the consistency of said application running on said given machine.

Fig. 1.

09/380250



Return To :

Tuxedo Commands

Start/Stop Tuxedo Configuration

Set up a Tuxedo Application

Manage the Listener Processes

Start Listener Processes

Stop Listener Processes

Change/Show Listener Process Parameters

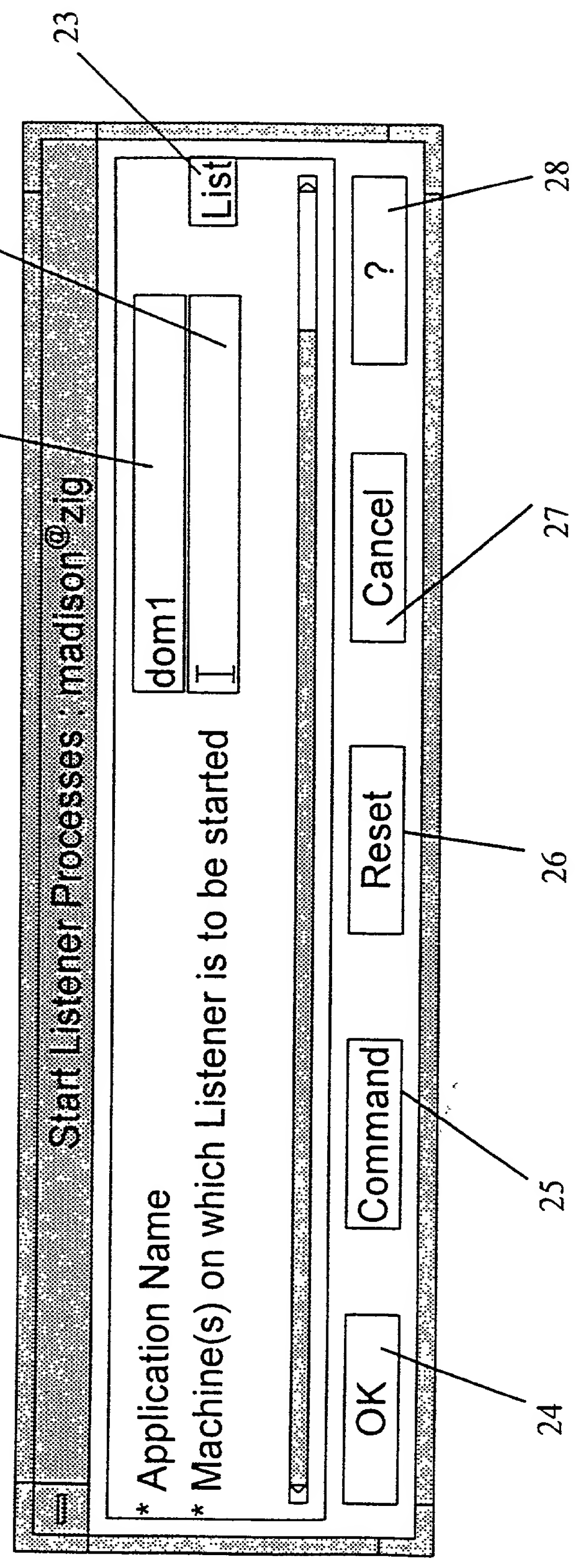
Show currently running Listener Processes

Check consistency of Listener Process scripts with TU

Update Listener Process scripts to TUXCONFIG Level

FIG. 1

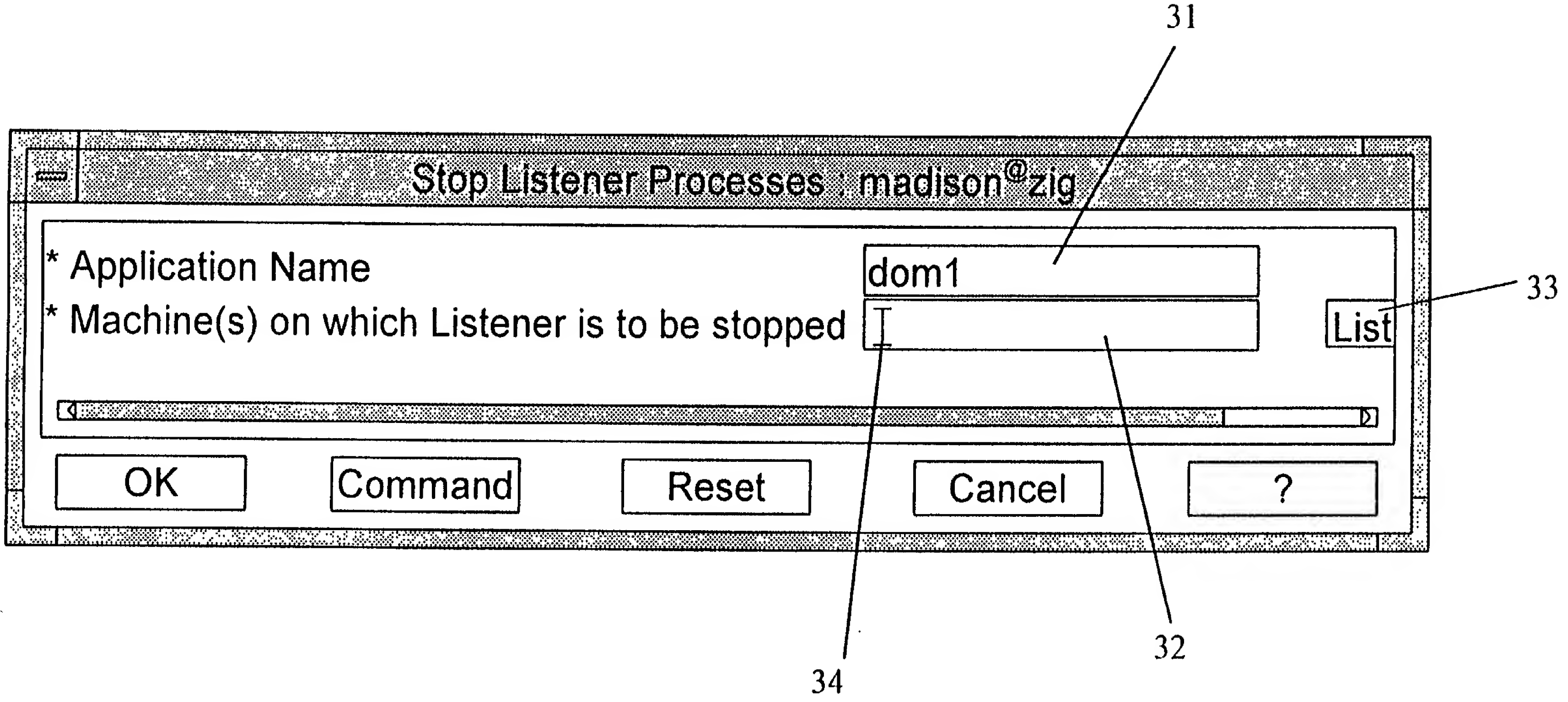
09/380250


Start Listener Processes : madison@zig

* Application Name dom1

* Machine(s) on which Listener is to be started

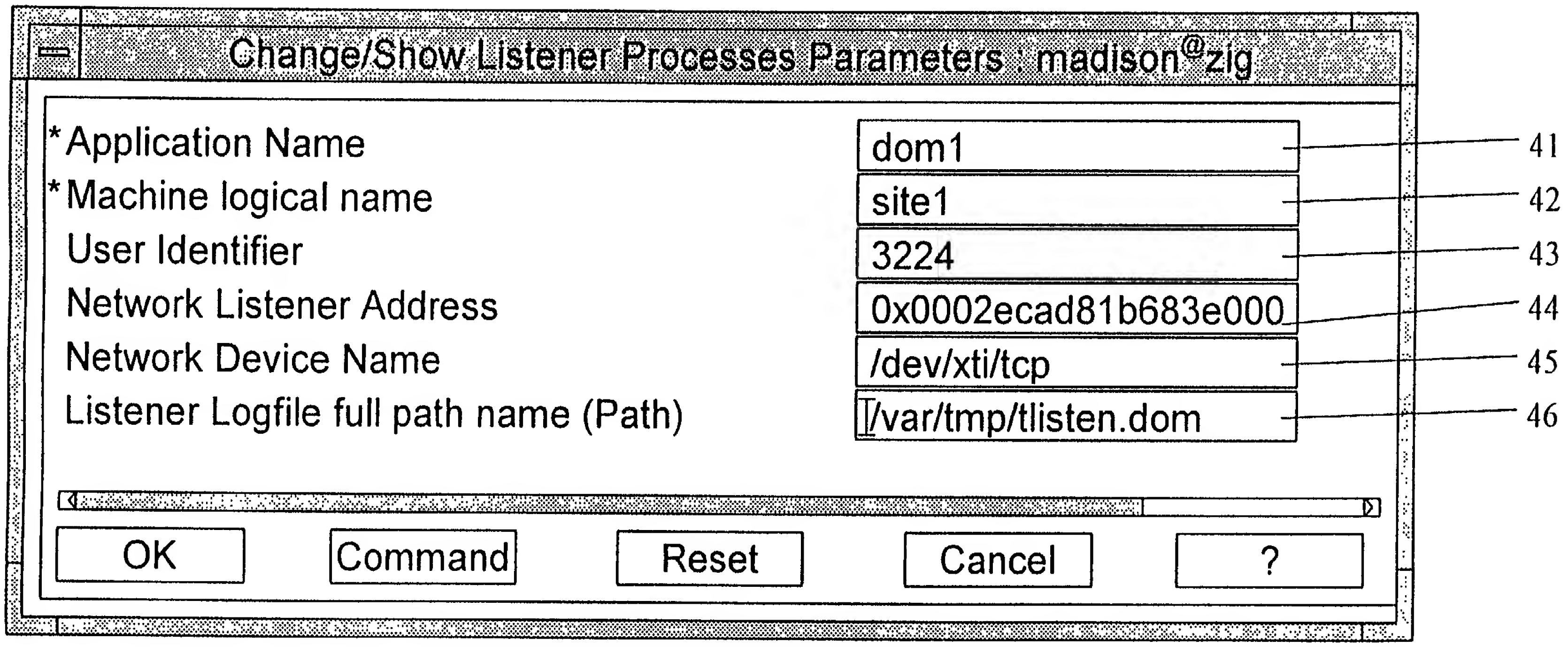
FIG. 2



Stop Listener Processes : madison@zig

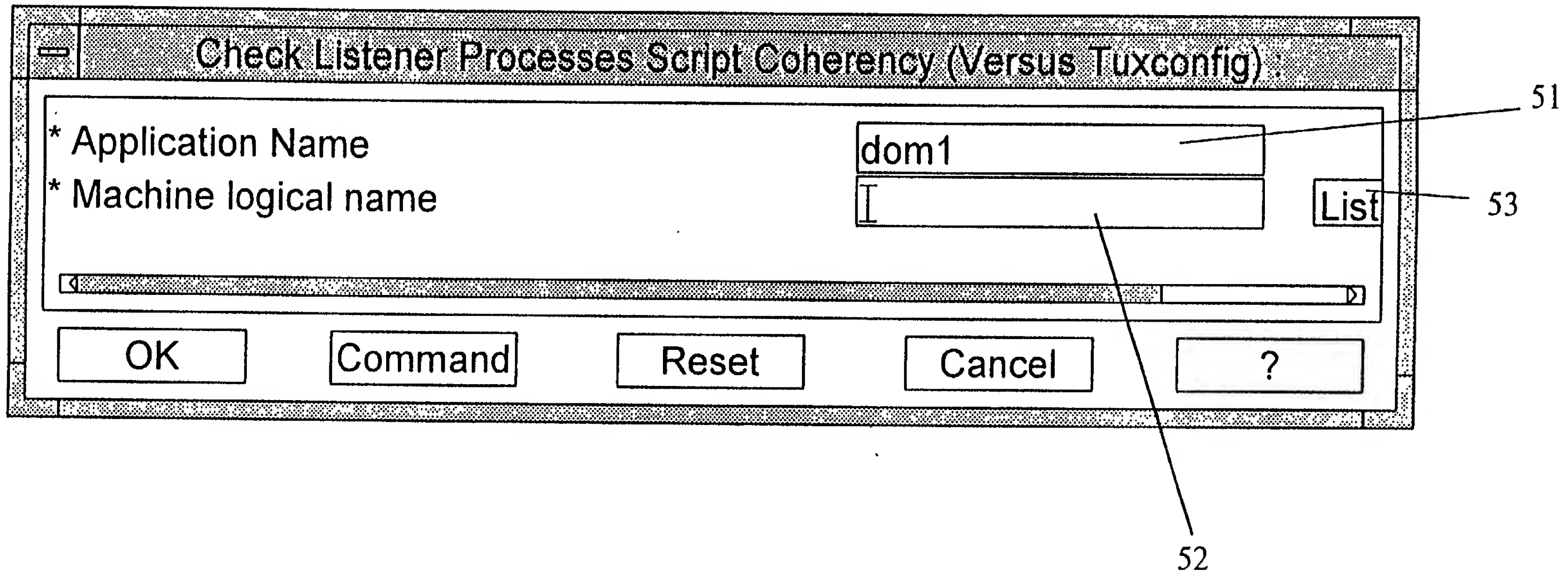
* Application Name: dom1
* Machine(s) on which Listener is to be stopped:

FIG. 3


Change/Show Listener Processes Parameters : madison@zig

* Application Name	dom1
* Machine logical name	site1
User Identifier	3224
Network Listener Address	0x0002ecad81b683e000
Network Device Name	/dev/xti/tcp
Listener Logfile full path name (Path)	/var/tmp/tlisten.dom

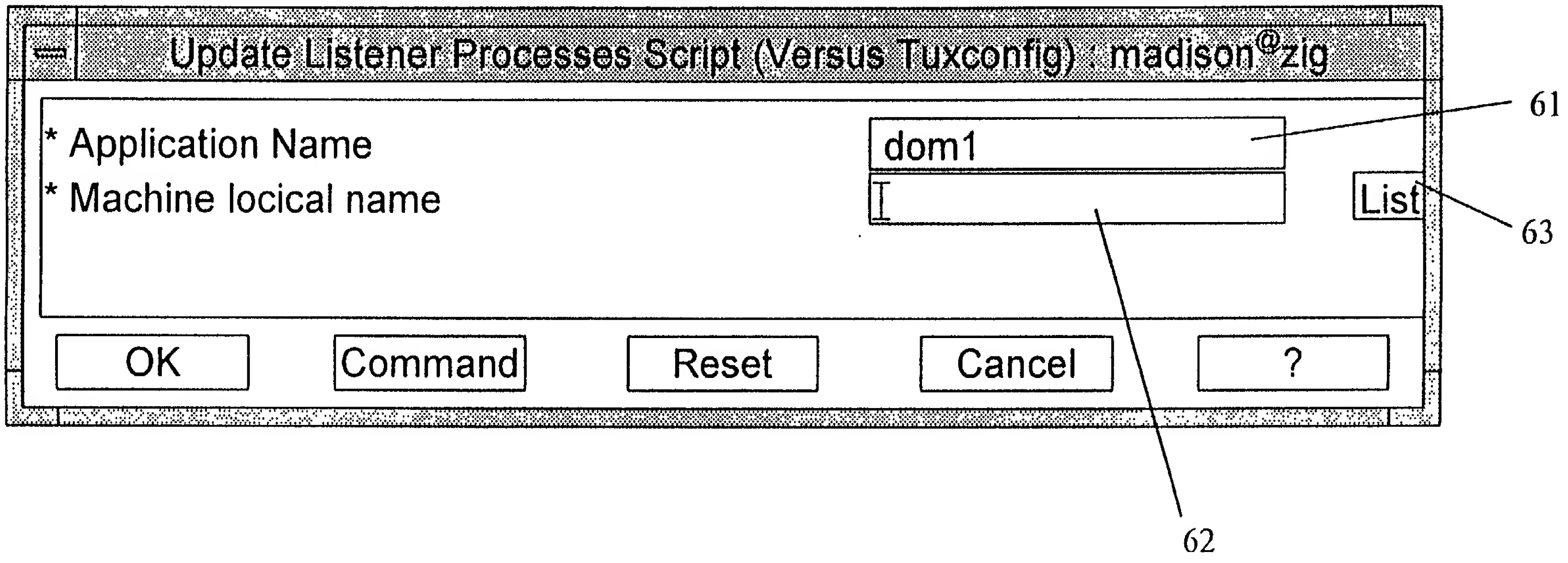
FIG. 4


Check Listener Processes Script Coherency (Versus Tuxconfig) :

* Application Name dom1

* Machine logical name

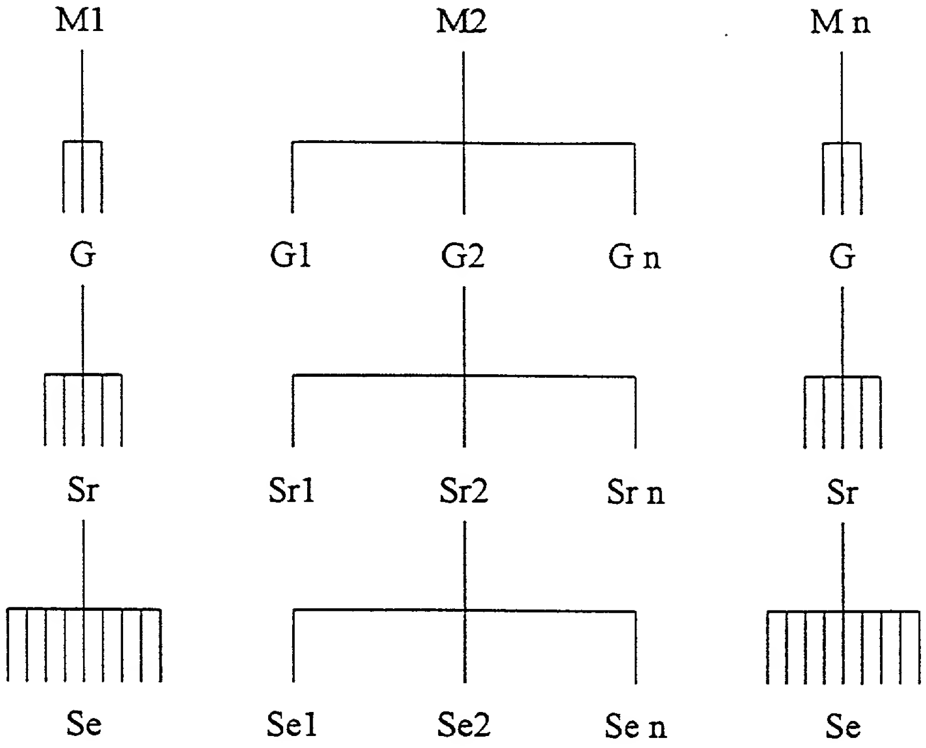
FIG. 5


Update Listener Processes Script (Versus Tuxconfig) : madison@zig
61
63
62
List
OK
Command
Reset
Cancel
?

* Application Name dom1

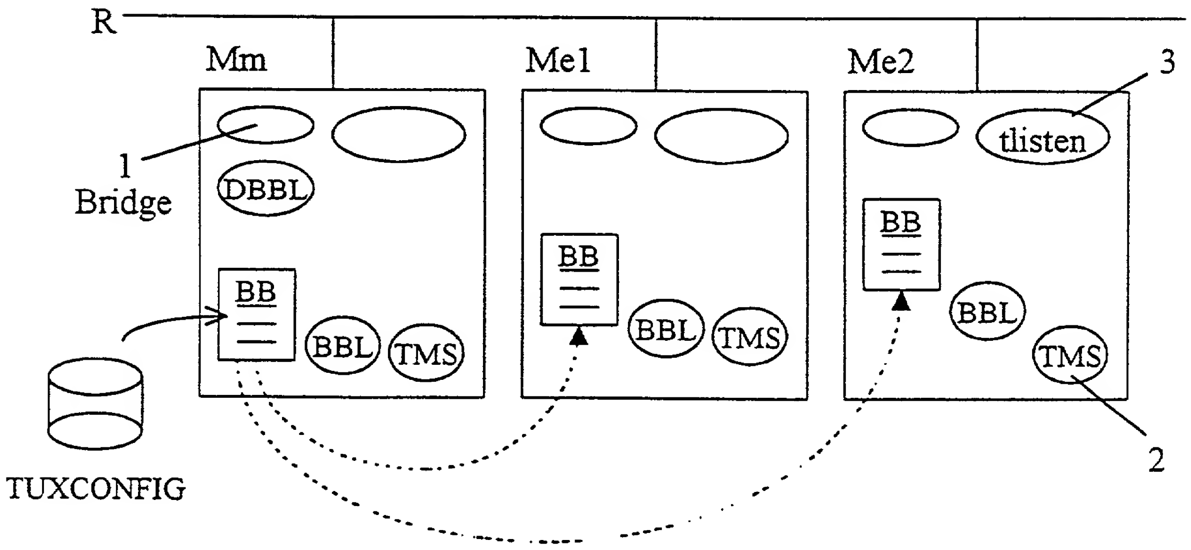
* Machine locical name

FIG. 6



M1
M2
M n
G
G1
G2
G n
G
Sr
Sr1
Sr2
Sr n
Sr
Se
Se1
Se2
Se n
Se

FIG. 7



R
Mm
Me1
Me2
3
tlisten
1
Bridge
DBBL
BB
BBL
TMS
BB
BBL
TMS
BB
BBL
TMS
2
TUXCONFIG

FIG. 8

US 3587 HD

Declaration and Power of Attorney For Patent Application
Declaration Pour Demandes de Brevets Avec Pouvoirs

French Language Declaration

En tant qu' inventeur nomme ci-après, Je déclare par le présent acte que:

As a below named inventor, I hereby declare that:

Mon nom, mon domicile, mon adresse postale, ma nationalité sont ceux qui figurent ci-après,

My residence, post office address and citizenship are as stated below next to my name,

Je déclare que je crois être l'inventeur original, premier et unique (si un seul nom figure sur le présent acte) ou un des co-inventeurs, originaux et premiers (si plusieurs noms figurent sur le present acte) du sujet revendiqué et pour liquel un brevet est demande sur la base de l'invention intitulée:

Procédé dassistance à l'administration d'une application distribuée basée sur un fichier binaire de configuration dans un système informatique.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

______________________________,

dont la description (cocher la case correspondante)

[✓] est annexée au présent acte.

☐ a été déposée ______________

Numéro de série de la demande ______________

et modifiée le ______________ (si approprié)

the specification of which (check one)

☐ is attached hereto.

☐ was filed on ______________ as

Application Serial No. ______________

and was amended on ______________ (if applicable)

Je déclare par le présent acte avoir examiné et compris le contenu de la description identifiée ci-dessus, revendications y compris, et le cas échéant telle que modifiée par l'amendment cité plus haut.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

Je reconnais le devoir de divulguer l'information qui est en rapport avec l'examen de cette demande selon Titre 37 du Code des Reglements Fédéraux §1.56(a).

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

US 3587 HD

French Language Declaration

Je revendique par le présent acte le bénéfice de priorité étrangère selon Titre 35, du Code des Etats-Unis, §119 de toute demande de brevet ou d'attestation d'inventeur énumérée ci-après, et j'ai identifié également ci-après toute demande étrangère de brevet ou d'attestation d'inventeur ayant une date de dépôt antérieure à celle de la demande pour laquelle la priorité est revendiquée.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior foreign applications

Demande(s) de brevet anterieure(s) dans un autre pays:

(Number) (Numéro)	(Country) (Pays)	(Day/Month/Year Filed) (Jour/Mois/Année de dépôt)	Priority claimed / Droit de priorité revendiqué
97 16699	FRANCE	30.12.1997	[✓ Yes Oui] No Non
			Yes Oui / No Non
			Yes Oui / No Non

Je revendique par le présent acte, le bénéfice selon Titre 35 du Code des Etats-Unis, §120 de toute(s) demande(s) américaines énumérée(s) ci-après et; dans la mesure où le sujet de chacune des revendications de cette demande n'est pas divulgué dans la demande américaine antérieure, de la façon dêfinie par le premier paragraphe de Titre 35 du Code des Etats-Unis, §112, je reconnais le devoir de divulguer l'information pertinente selon Titre 37 du Code des Réglements Fédéraux, §1.56(a), toute information qui se présente entre la date de dépôt de la demande antérieure et la date de dépôt de la demande, soit nationale, soit internationale PCT.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.) (No. de Demande)	(Filing Date) (Date de Dépôt)	(Etat) (brevetée, pendante, abandonné)	(Status) (patented, pending, abandoned)

Je déclare par le présent acte que toutes mes déclarations, à ma connaissance, sont vraies et que toutes les déclarations faites à partir de renseignements ou de suppositions, sont tenues pour être vraies; de plus, toutes ces declarations ont été faites en sachant que de fausses déclarations volontaires u autres actes de même nature sont sanctionées par une amende ou un emprisonnement, ou les deux, selon la Section 1001, du Titre 18 de Code des Etats-Unis et que de selles déclarations délibérément fausses peuvent compromettre la validité de la demande ou du brevet délivré.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

US 3587 HD

French Language Declaration

POUVOIR: En tant qu'inventeur, je désigne l'(les) avocat(s) et/ou l'(les) agent(s) suivant(s) pour poursuivre la procédure de cette demande et traiter toute affaire la concernant supris du Bureau des Brevets et de Marques:

5

Harold L. Stowell, Reg. 17,233
Edward J. Kondracki, Reg. 20,604
Dennis P. Clarke, Reg. 22,549
William L. Feeney, Reg. 29,918
John C. Kerins, Reg. 32,421

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (*list name and registration number*)

Harold L. Stowell, Reg. 17,233
Edward J. Kondracki, Reg. 20,604
Dennis P. Clarke, Reg. 22,549
William L. Feeney, Reg. 29,918
John C. Kerins, Reg. 32,421

Adresser toure correspondance à:

Edward J. Kondracki, Esq.
KERKAM, STOWELL, KONDRACKI
& CLARKE, P.C.
5203 Leesburg Pike, Suite 600
Falls Church, VA 22041

Send Correspondence to:

Edward J. Kondracki, Esq.
KERKAM, STOWELL, KONDRACKI
& CLARKE, P.C.
5203 Leesburg Pike, Suite 600
Falls Church, VA 22041

Adresser toute communication téléphonique à: (Nom) (*Numéro de téléphone*)

Edward J. Kondracki, Esq.
(703) 998-3302

Direct Telephone Calls to: (*name and telephone number*)

Edward J. Kondracki, Esq.
(703) 998-3302

1-00

Nom complet du seul ou premier inventeur Baillif Christian	Full name of sole or first inventor
Signature de l'inventeur [illegible] Date 4 Fevrier 1998	Inventor's signature Date
Domicile 7 bis, avenue du Petit Chambord, Bourg la Reine, France	Residence FRX
Nationalité Française	Citizenship
Adresse Postale 7 bis, avenue du Petit Chambord, Bourg la Reine,	Post Office Address
France	

2-00

Nom complet du second co-inventeur, le cas echeant Dia Mama Saidou	Full name of second joint inventor, if any
Signature de l'inventeur [illegible] Date 4 Février 1998	Second Inventor's signature Date
Domicile 181, avenue Jean Jaurès, 92290 Chatenay Malabry, France	Residence FRX
Nationalité Française	Citizenship
Adresse Postale 181, avenue Jean Jaurès, 92290 Chatenay Malabry,	Post Office Address
France	

(Fournir les mêmes renseignements et la signature de tout co-inventeur supplémentaire.)

(Supply similar information and signature for third and subsequent joint inventors.)